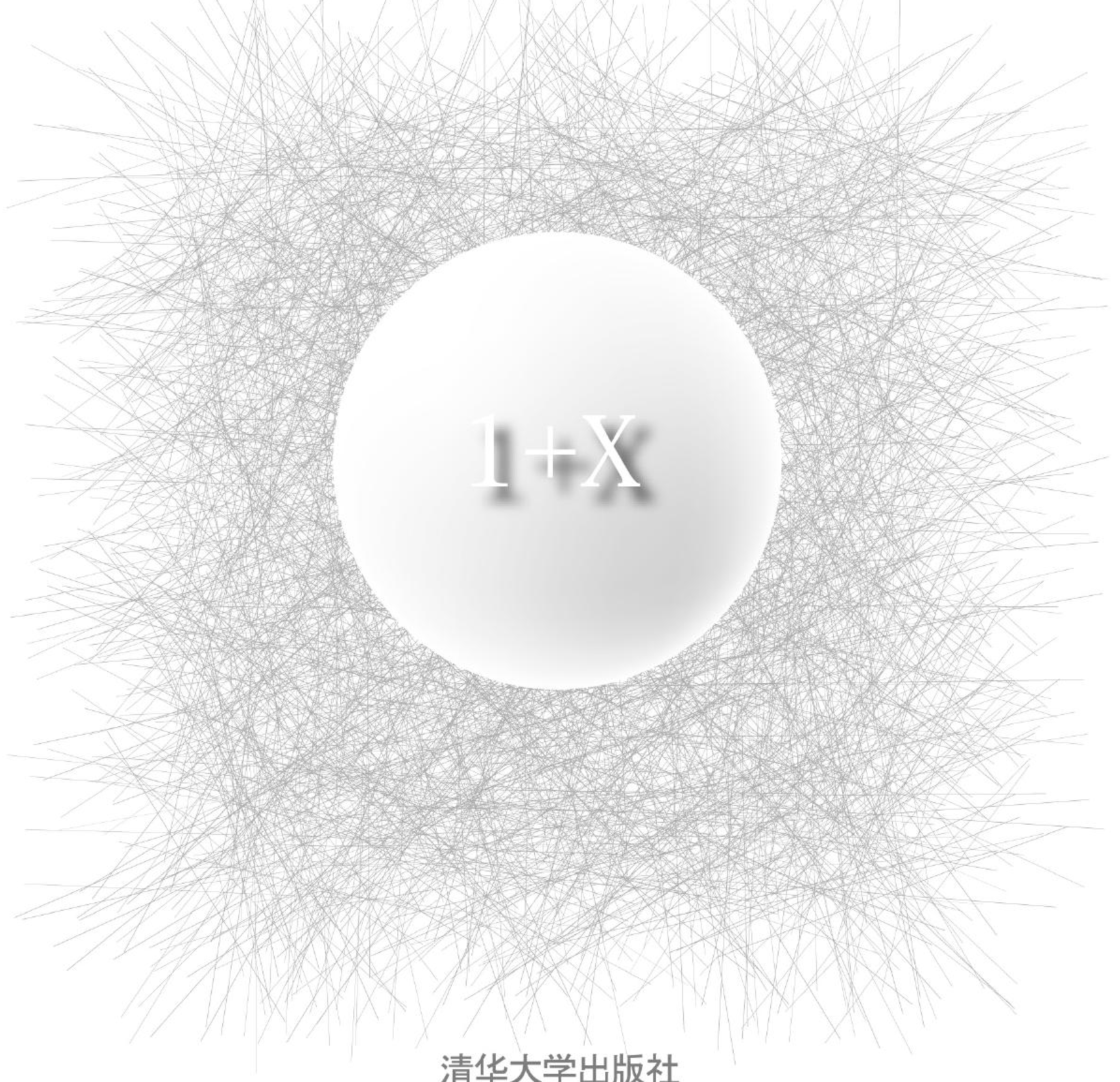
西安交通大学本科“十二五”规划教材
西安交通大学“985”工程三期重点建设实验系列教材

大学计算机基础教育规划教材

# C++程序设计实验指导与习题解析

赵英良　卫颜俊　仇国巍　编著
冯博琴　审




清华大学出版社

大学计算机基础教育规划教材
西安交通大学本科“十二五”规划教材
西安交通大学“985”工程三期重点建设实验系列教材

# C++ 程序设计实验指导与习题解析

赵英良　卫颜俊　仇国巍　编著
冯博琴　审

清华大学出版社
北京

## 内 容 简 介

本书是与《C++ 程序设计教程》配套的实验指导和习题解析。全书共分四部分，分别为环境的使用、实验指导、习题解析和常用资料。环境的使用部分包括 Visual C++ 6.0、Visual C++ 2010、C++ Builder 6.0 的使用和跟踪、调试程序的方法、帮助的使用方法等内容；实验指导部分提供 11 个实验、53 道题目，从问题分析、算法描述、编程指南、测试指南、问题扩展等方面指导学生完成实验；习题解析部分对《C++ 程序设计教程》中的部分习题进行了分析，给出了问题分析、算法描述、编程提示、测试指南、问题扩展等方面的指导；常用资料部分给出了常见词汇、常见编译错误、常用库函数等实验过程中需要查阅的资料。

本书内容丰富、实用，指导切实、及时；既可作为高等学校计算机程序设计课程的实验用书，也可供程序设计爱好者和相关工程技术人员参考。



**图书在版编目（CIP）数据**

C++ 程序设计实验指导与习题解析/赵英良等编著. --北京：清华大学出版社，2013
大学计算机基础教育规划教材
ISBN 978-7-302-33058-5

Ⅰ. ①C… Ⅱ. ①赵… Ⅲ. ①C 语言－程序设计－高等学校－教学参考资料 Ⅳ. ①TP312

中国版本图书馆 CIP 数据核字（2013）第 145943 号

**责任编辑**：焦　虹
**封面设计**：傅瑞学
**责任校对**：时翠兰
**责任印制**：刘海龙

**出版发行**：清华大学出版社
**网　　址**：http://www.tup.com.cn，http://www.wqbook.com
**地　　址**：北京清华大学学研大厦 A 座　　**邮　　编**：100084
**社 总 机**：010-62770175　　**邮　　购**：010-62786544
**投稿与读者服务**：010-62776969，c-service@tup.tsinghua.edu.cn
**质 量 反 馈**：010-62772015，zhiliang@tup.tsinghua.edu.cn
**课 件 下 载**：http://www.tup.com.cn，010-62795954
**印 装 者**：三河市李旗庄少明印装厂
**经　　销**：全国新华书店
**开　　本**：185mm×260mm　　**印　　张**：10.75　　**字　　数**：249 千字
**版　　次**：2013 年 9 月第 1 版　　**印　　次**：2013 年 9 月第 1 次印刷
**印　　数**：1～2000
**定　　价**：23.00 元

---

产品编号：055007-01

# 前　言

程序设计是一门实践性很强的课程。学习程序设计不仅要了解语言的语法,重要的是通过多练来学习计算机解决问题的思路和方法。本书是与《C++ 程序设计教程》配套的辅导教材。

本书的宗旨是为 C++ 程序设计课程实验提供切实的指导。本书的特色有:

(1) 实用的内容。本书从解决学生在实验中遇到的问题出发,介绍编辑、编译、连接和运行程序的方法,以及调试方法和技巧。书中资料来源于实验过程中同学们常提的问题。

(2) 切实的指导。同学们常反映的问题是:别人的程序能看懂,自己就不知怎样写了。本书加强了问题分析、算法描述和编程提示的指导内容,希望同学们学会分析问题的方法和解决问题的步骤。

(3) 良好的习惯。本书用规范的格式编写算法,输入输出要求意义明确、功能清楚,程序中应有适当的注释,要对程序进行多种角度的测试。希望同学们能养成良好的编程习惯。

(4) 积极地思考。本书在实验题目后面会给出一些思考题、练习题,并在适当的地方给出与程序有关的趣味故事,引导学生去探索更深刻的问题,培养编程的乐趣,发现编程的奥妙。

本书分为四部分。

第 1部分是环境的使用,包括 Visual C++ 6.0(简称 VC6)、Visual C++ 2010(简称 VS2010)和 C++ Builder 6.0三种环境的使用。这部分的内容不需单独实验,需要时查阅即可。这部分包括了同学们在环境使用中常见的问题。例如:如何解决编译错误,如何单步跟踪程序,如何设置断点,如何在一个工程中编写多个程序等。

第 2部分是实验指导。这部分内容与教材的内容相对应,共 11 个实验。每个实验包含 3~8个题目,每个题目从问题分析、算法描述、编程提示、测试指南、问题扩展等方面给出指导。教师可以选择安排在两小时的上机时间内完成 2~3个题目,其他题目在课外完成。5、6、7三个实验可以分两次(4学时)上机完成。

第 3部分是习题解析,其中给出了对《C++ 程序设计教程》中绝大部分习题的分析供参考。每个题目的解析内容与实验指导类似,简单的题目没有解析。

第 4部分是常用资料。这部分是同学们实验中经常需要查阅的内容,如 ASCII 编码表,编译、连接常见的英文词汇、错误信息,常用的库函数等。

本书内容丰富、实用,指导切实、及时,是指导而不是答案。本书可作为高等学校计算

机程序设计课程的实验用书，也可供程序设计爱好者和相关工程技术人员参考。

本书由赵英良主编，冯博琴教授审阅。实验1～4和习题1～4的解析由赵英良编写，实验5、7、8、9和习题5、7、8、9的解析由卫颜俊编写，实验6、10、11和习题6、10、11的解析由仇国巍编写。本书获得2012年西安交通大学实验教材立项和资助，是西安交通大学本科“十二五”规划教材和西安交通大学“985”工程三期重点建设实验系列教材。本书得到了西安交通大学教务处徐忠锋副处长、冯晓娟等老师以及西安交通大学计算机教学实验中心同事的帮助和支持，在此一并表示感谢；此外，也向参考文献的作者表示感谢。

由于作者学识浅陋，加之时间仓促，书中可能会有不少错误和不当之处，恳请读者批评指正。

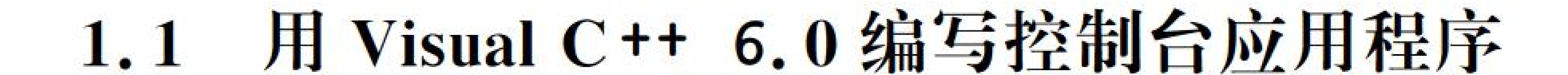
编者

# 目 录

C++程序设计实验指导与习题解析

# 第1部分

## 环境的使用



## 1.1 用 Visual C++ 6.0 编写控制台应用程序

尽管 Visual C++ 已经升级了很多版本,但对控制台应用程序的编写而言,Visual C++ 6.0 仍不失为一个简洁方便的编程环境。就控制台 C++ 程序来说,它与其他后续版本的功能没有差别。

### 1.1.1 进入和退出 Visual C++ 集成开发环境

#### 1. 启动并进入 Visual C++ 集成开发环境

通常有三种方法。

(1) 使用"开始"菜单。选择"开始"菜单中的"程序",然后选择 Microsoft Visual Studio 6.0 级联菜单,再选择 Microsoft Visual C++ 6.0 菜单项,如图 1-1 所示。

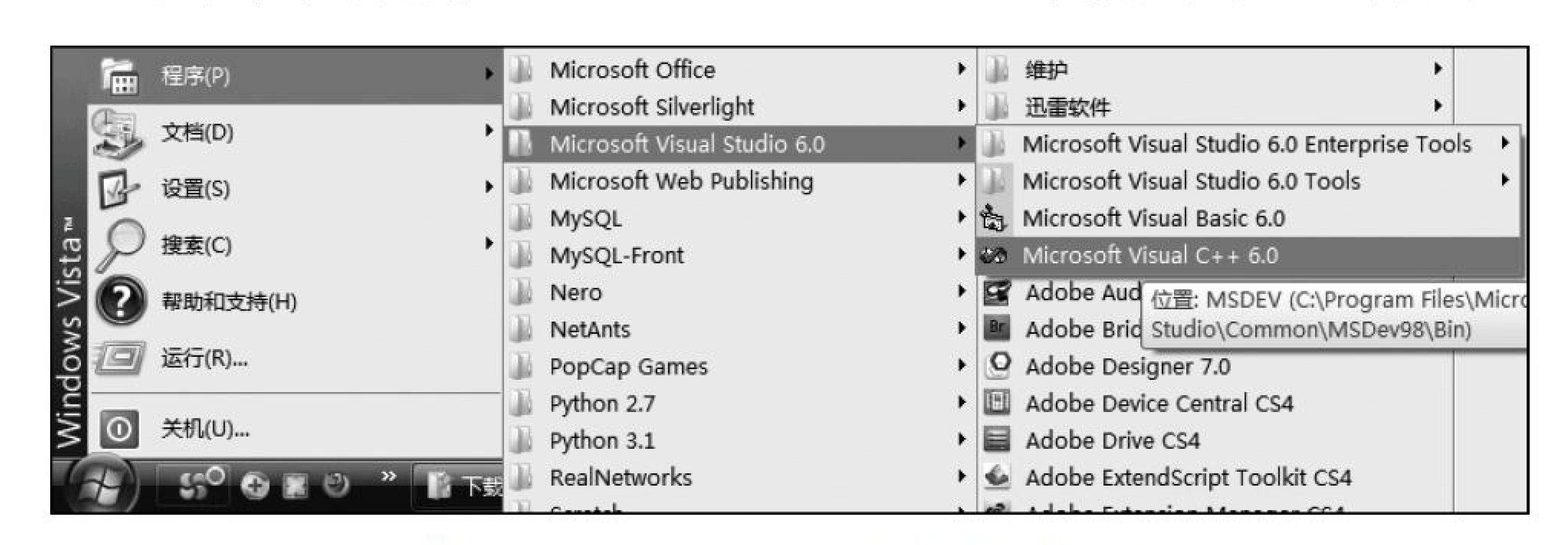



图 1-1 启动 Visual C++ 6.0 集成开发环境

(2) 使用快捷方式图标。在桌面上创建快捷方式,直接双击快捷方式图标(见图 1-2)。




图 1-2 Visual C++ 6.0 快捷方式图标

(3) 双击 Visual C++ 6.0 工程文件名。Visual C++ 6.0 的工程文件扩展名为 dsw。如果已经创建过 Visual C++ 6.0 的工程,在工程文件夹中双击扩展名为 dsw 的工程文件,可以启动 Visual C++ 6.0 并打开工程文件。

### 2. 退出 Visual C++ 6.0 集成开发环境

使用集成环境中的 File|Exit 菜单项或单击右上角的关闭按钮。

## 1.1.2 创建工程、打开已有工程

### 1. 创建工程

(1) 选择"开始"|"程序"|Microsoft Visual Studio 6.0|Microsoft Visual C++ 6.0 启动 Visual C++ 6.0 集成开发环境(见图 1-3)。

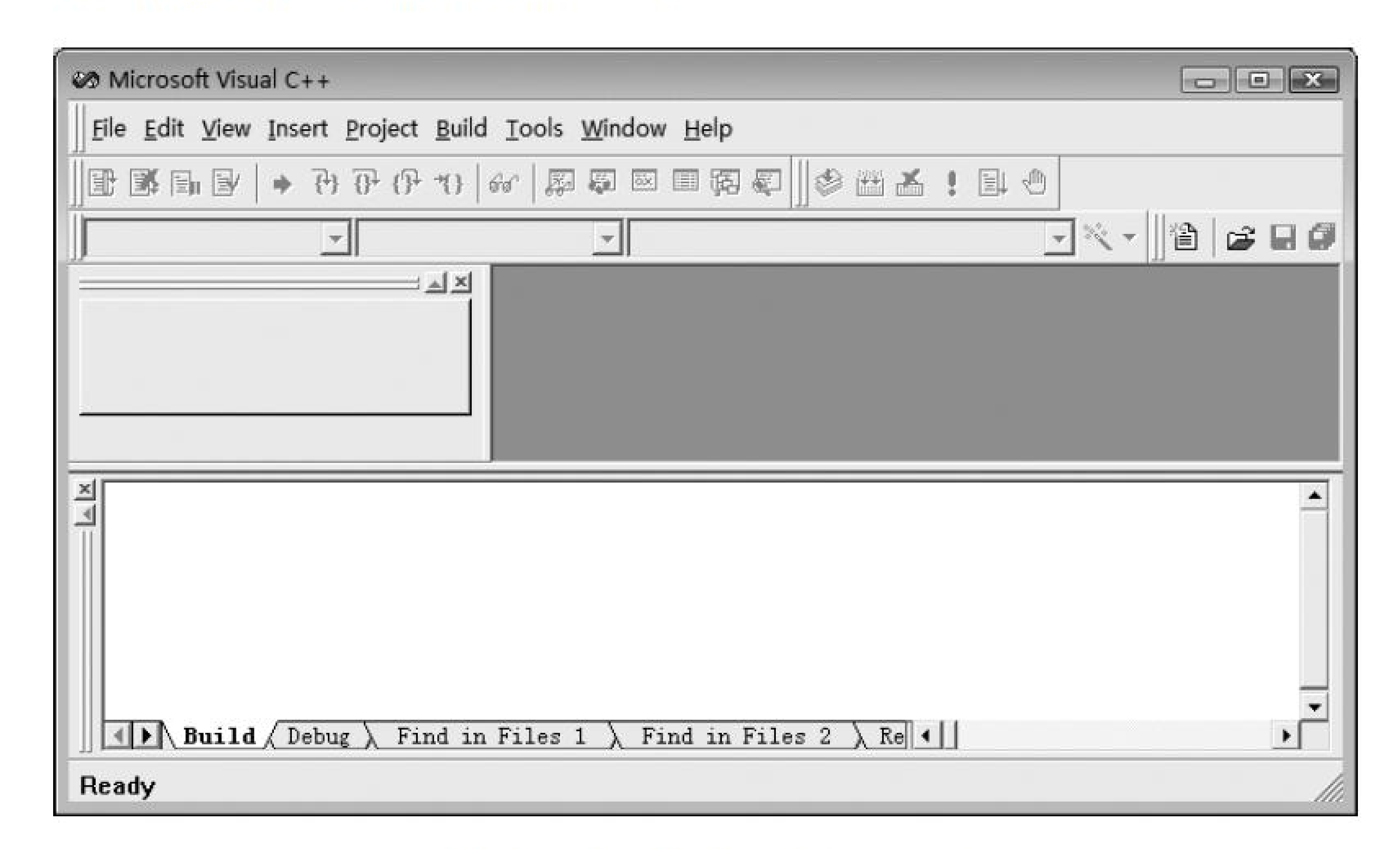



图 1-3 启动 Visual C++ 6.0

(2) 在集成环境中执行 File|New 菜单命令(见图 1-4)。在图 1-4 所示的对话框中选择工程类型为"Win32 Console Application"(Win32 控制台应用程序);选择工程的存放路径,如 d:\;输入工程名称,如 cpptmp2。然后单击 OK 按钮。注意,这时的标签是 Projects。

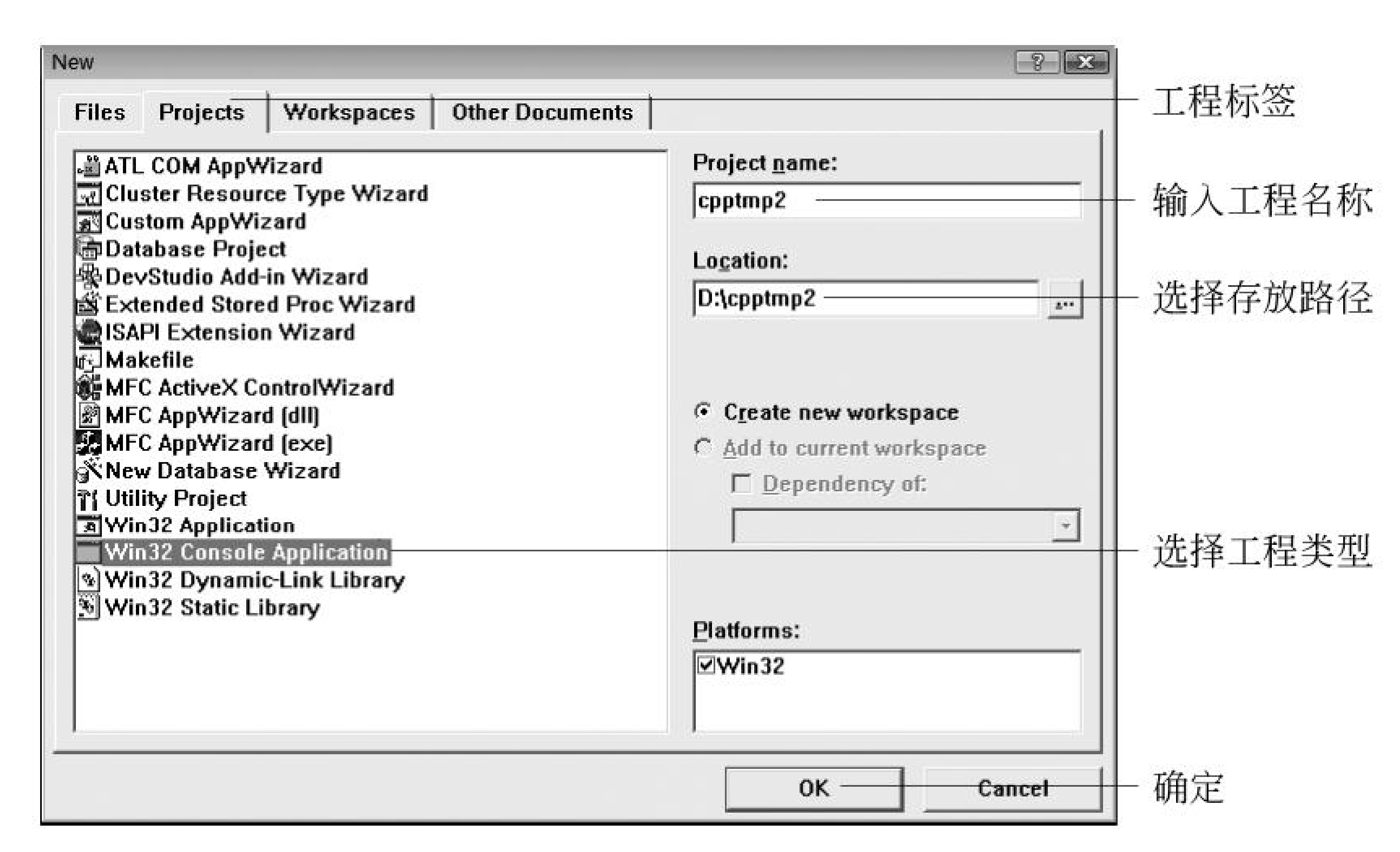



图 1-4 创建工程

(3) 在图1-5所示的对话框中选择 An empty project(空工程),单击 Finish 按钮。在接下来的 New Project Information(新工程信息)对话框中单击 OK 按钮,结果如图1-6所示。

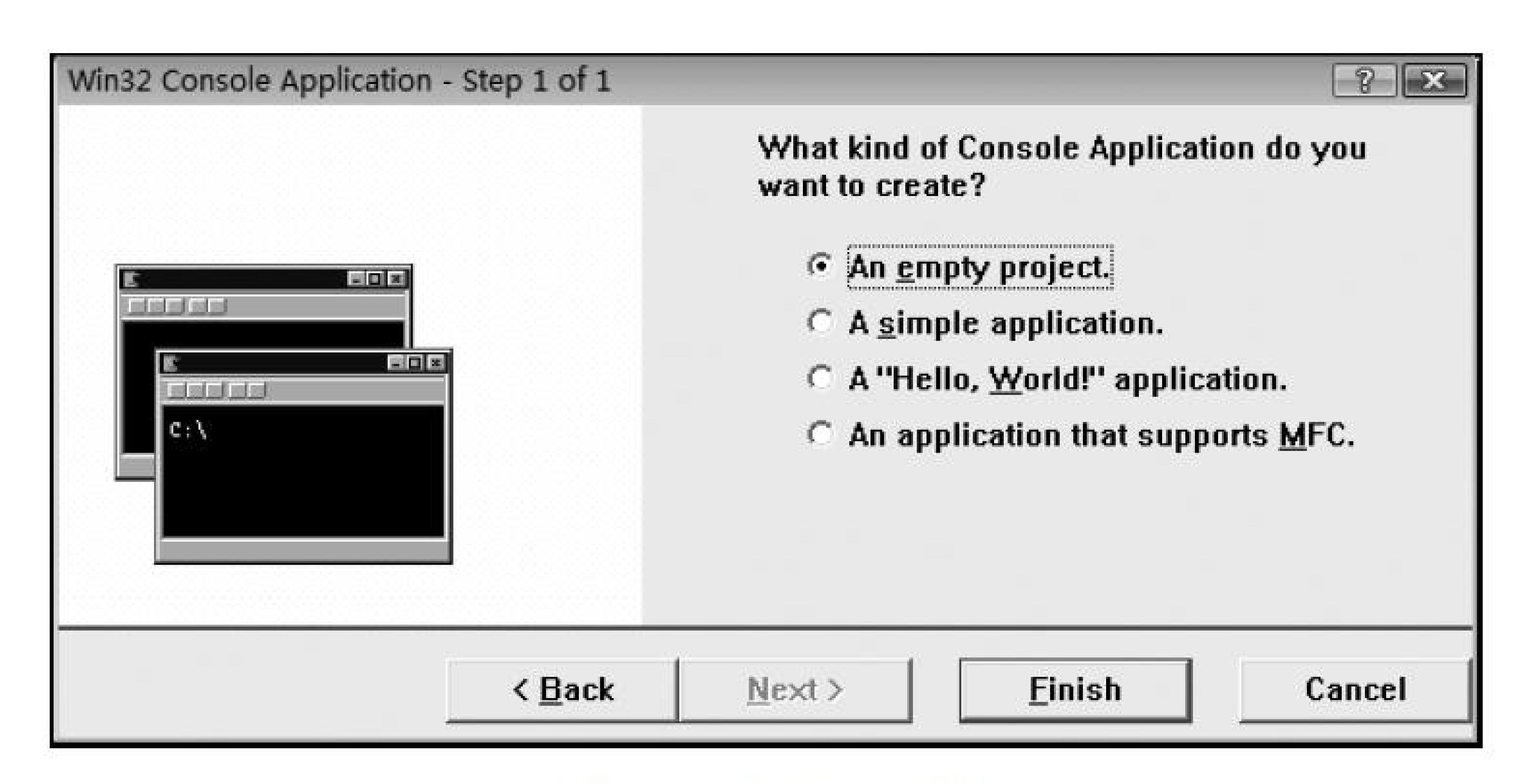



图1-5　选择空工程

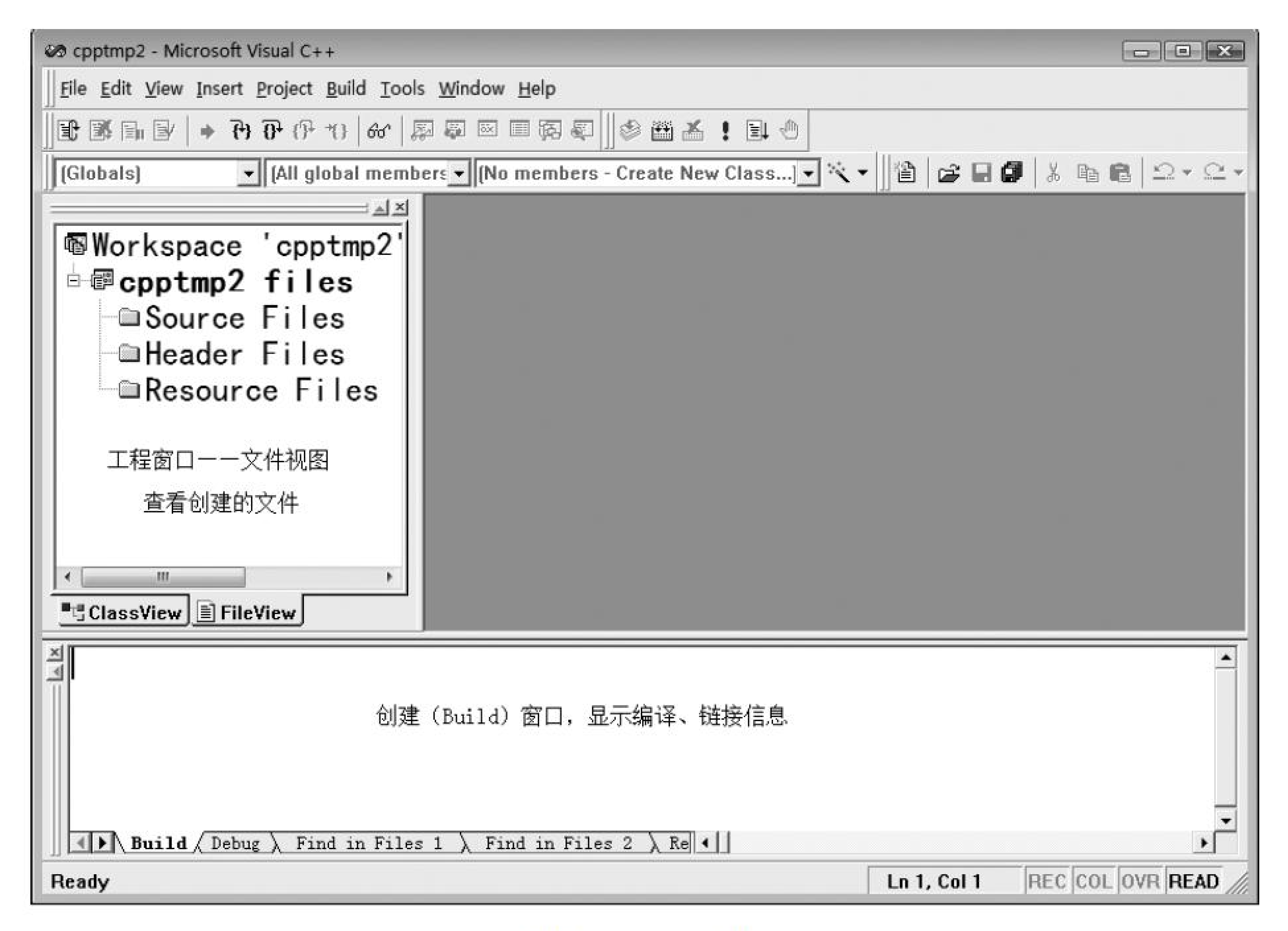



图1-6　空工程

工程创建后,在选择的路径下,会创建一个以工程名称命名的文件夹,其中有.dsw为扩展名的工程文件及其他相关文件(见图1-7)。以后编写的扩展名为cpp的程序文件也会在此文件夹下,备份时可以从此处复制。程序生成的扩展名为exe的可执行文件在debug文件夹中,它可以被单独复制到其他存储位置,单独执行,这是我们编写的软件。

(4) 保存工程。

选择 File|Save Workspace 即可。

**2. 打开已有工程**

选择 File|Open Workspace 或在工程文件夹(见图 1-7)中双击 *.dsw 文件(比如 cpptmp2.dsw)。

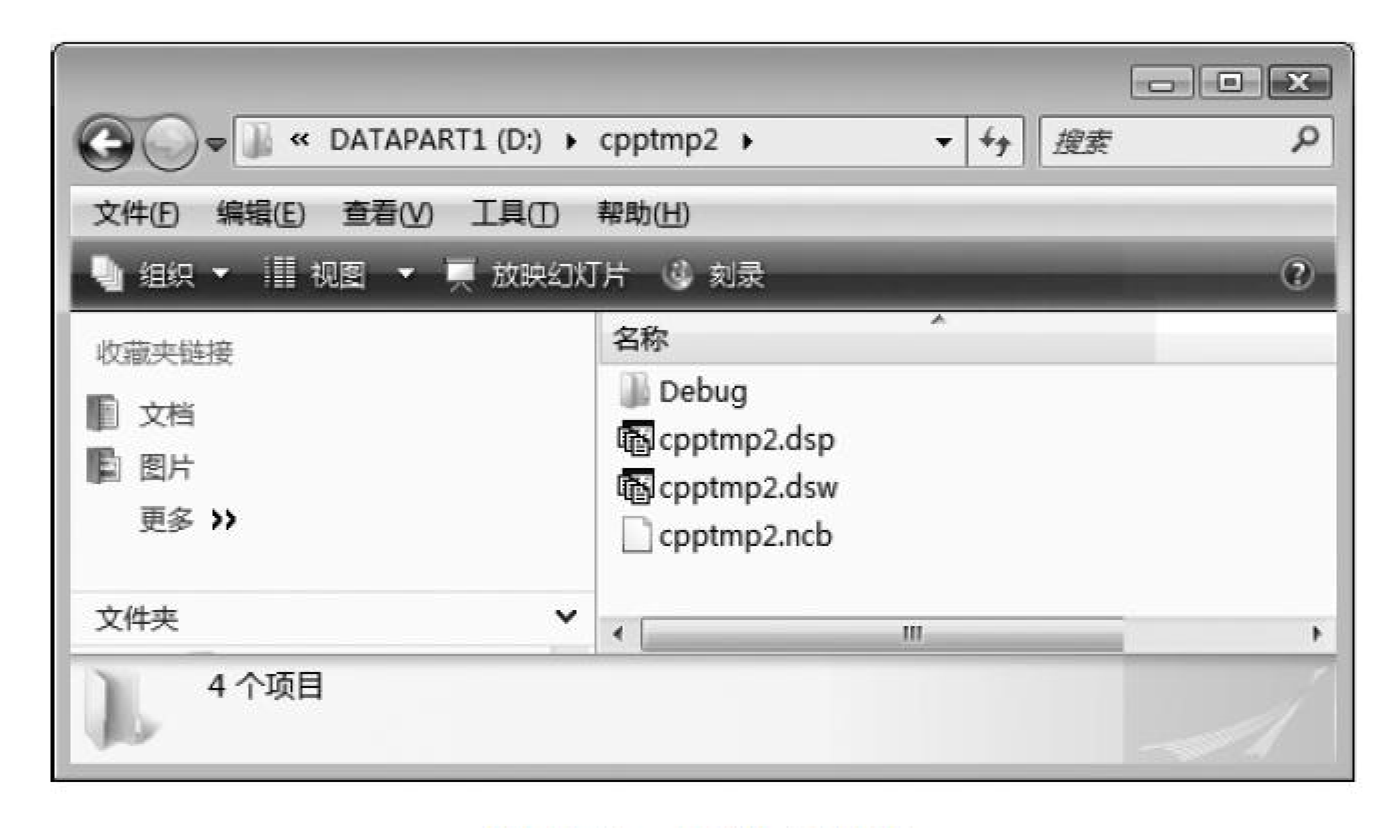



图 1-7 工程文件夹

## 1.1.3 创建 C++ 程序文件

**1. 编写程序**

(1) 启动 Visual C++ 6.0。

(2) 创建工程。

(3) 执行 File|New 菜单命令,打开 New 对话框。注意这次是 Files(文件)标签(见图 1-8)。选择文件类型为"C++ Source File";输入文件名,可以和工程名相同,也可以不同,单击确定按钮。

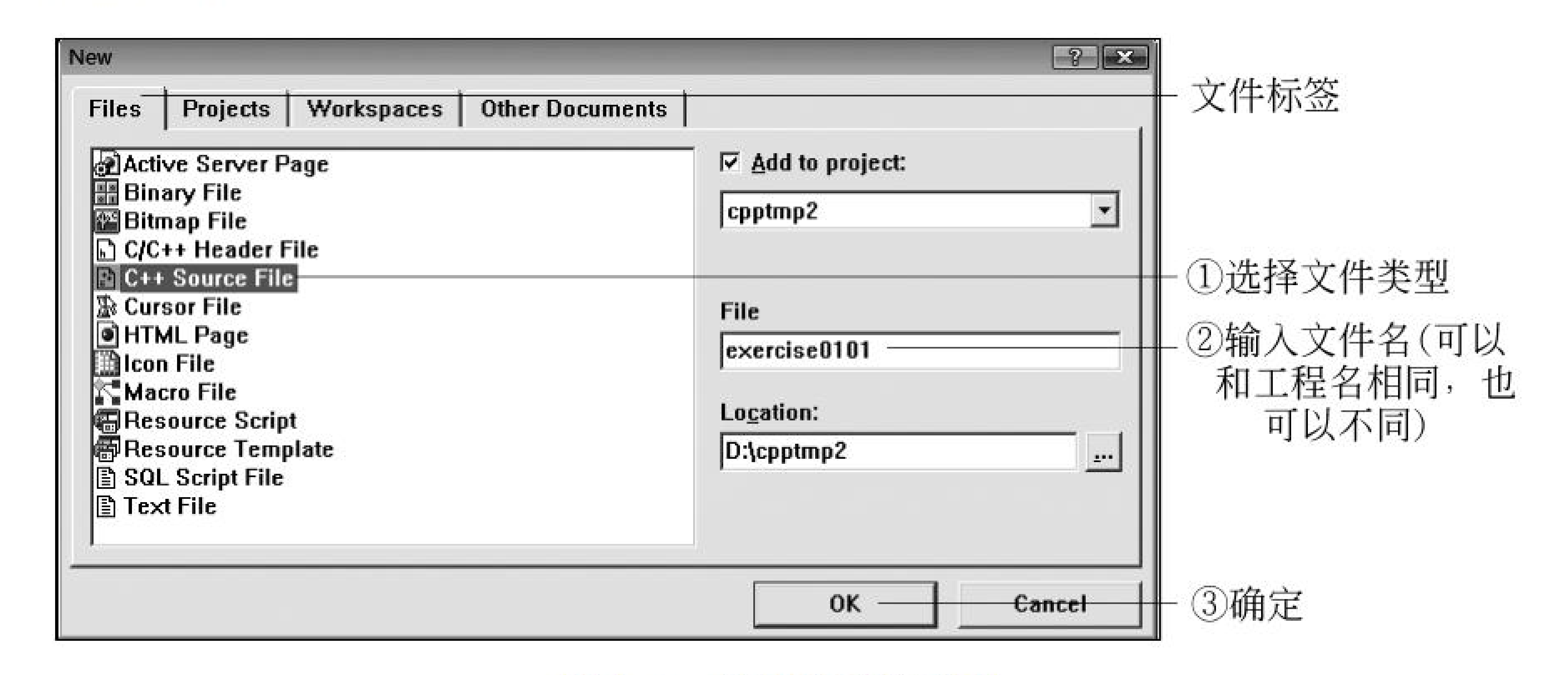



图 1-8 创建源程序文件

(4) 在集成环境右上部的文件窗口中输入、编辑源程序(见图 1-9)。

**2. 备份程序**

在 C++ 的学习过程中,编写的 C++ Source 文件最重要,它以 cpp 为扩展名存在于工

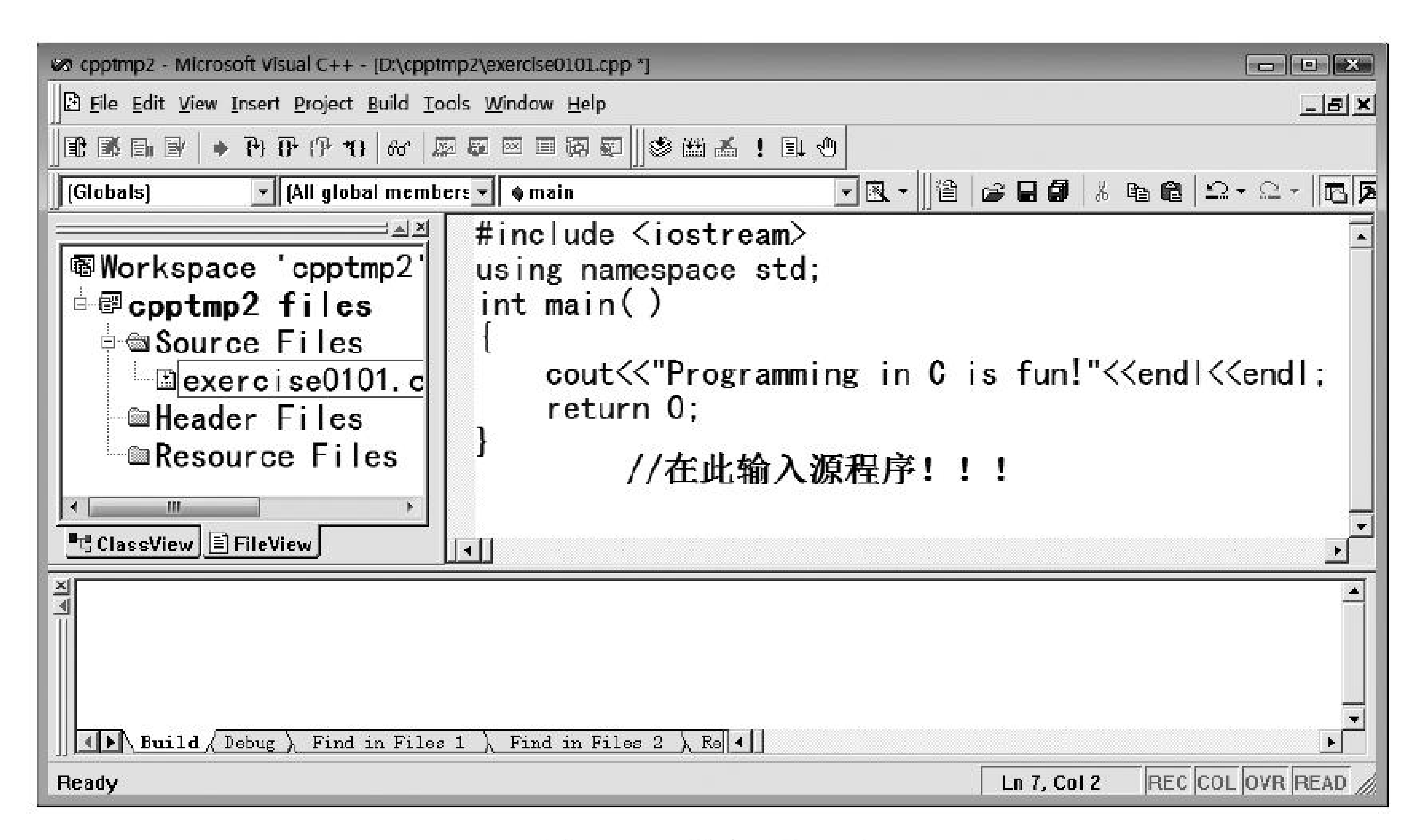



图 1-9 编辑源程序

程文件夹下，要保存，只要复制该文件即可，不必备份整个工程文件夹。

## 1.1.4 编译、连接、运行程序

### 1. “Build”菜单的“Compile …”编译程序

编译检查语法错误。如果有语法错误，则显示在下方的 Build 窗口中。双击错误提示，系统将错误所在的行在文件编辑窗口中指示出来(见图 1-10)。

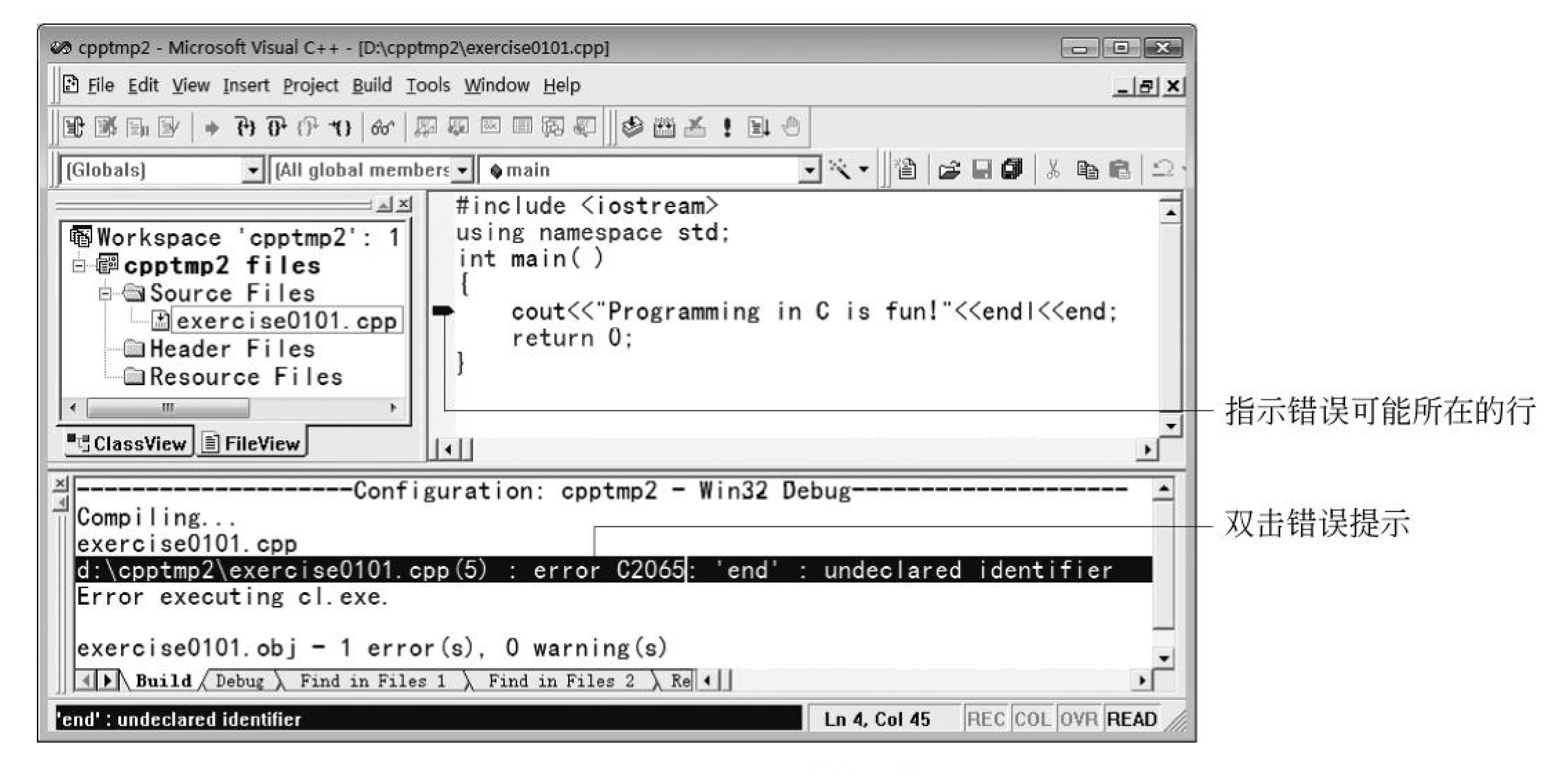



图 1-10 编译信息

**注意**：有多个错误时，不要着急，只双击第 1 个错误，修改后再次编译。一个语法错误可能引发很多条 error 信息，因此，不要一次修改多个错误，除非清楚地知道哪儿错了。最好修改一处，编译一次。因为修改一处有可能消除多条错误信息。

编译时的另一类信息是警告(warning)。警告信息一般是触发了 C/C++ 的自动规

则,如将一个单精度(浮点)型数据赋给整型变量,需要系统将单精度型数据自动转换为整型,此时小数部分会丢失,因而系统给出警告信息。

警告信息不会影响程序执行,但可能会使运行结果错误。

**2. "Build"菜单的"Build… "连接程序(生成可执行文件)**

连接是组装程序的过程。连接也会有错误,如缺少头文件,没有 main 函数,多个 main 函数等。连接错误也在 Build 窗口中列出,处理方法与编译错误相同。

**3. "Build"菜单的"Execute…"运行程序**

没有编译、连接错误后,就可以执行程序了。执行程序,完成程序设计的功能,如输入、计算、显示等。

执行程序,会出现一个命令提示符窗口(见图 1-11),在其中输入数据,显示结果。

编译、连接和运行也可以使用工具栏的按钮,见图 1-12。

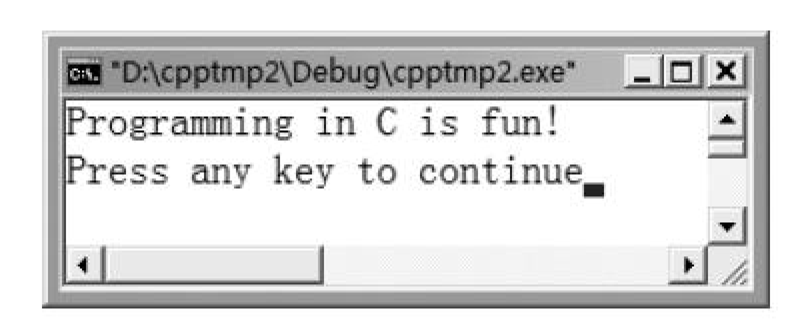



图 1-11 命令提示符窗口




图 1-12 编译、连接和运行的按钮

## 1.1.5 程序的跟踪调试

编译、连接没有错误,程序就可以运行了;但可以运行的程序,结果不一定正确。例如下面求两个数的最大值的程序:

```
#include <iostream>
using namespace std;
int main()
{
    double a,b,y;
    cin>>a>>b;
    if (a>b)
    {
        y=a;
    }
    else;
    y=b;
    cout<<y<<endl;
    return 0;
}
```

如果输入 4 5,结果是 5,正确;如果输入 5,4,结果为 4,不正确。这样的错误,称为逻辑错误。查找并纠正错误的过程称为调试。调试程序有以下一些方法。

**1. 单步跟踪**

在工具栏空白处右单击鼠标，在打开的快捷菜单中选择 Debug 菜单项（见图 1-13），打开 Debug 工具栏（见图 1-14）。

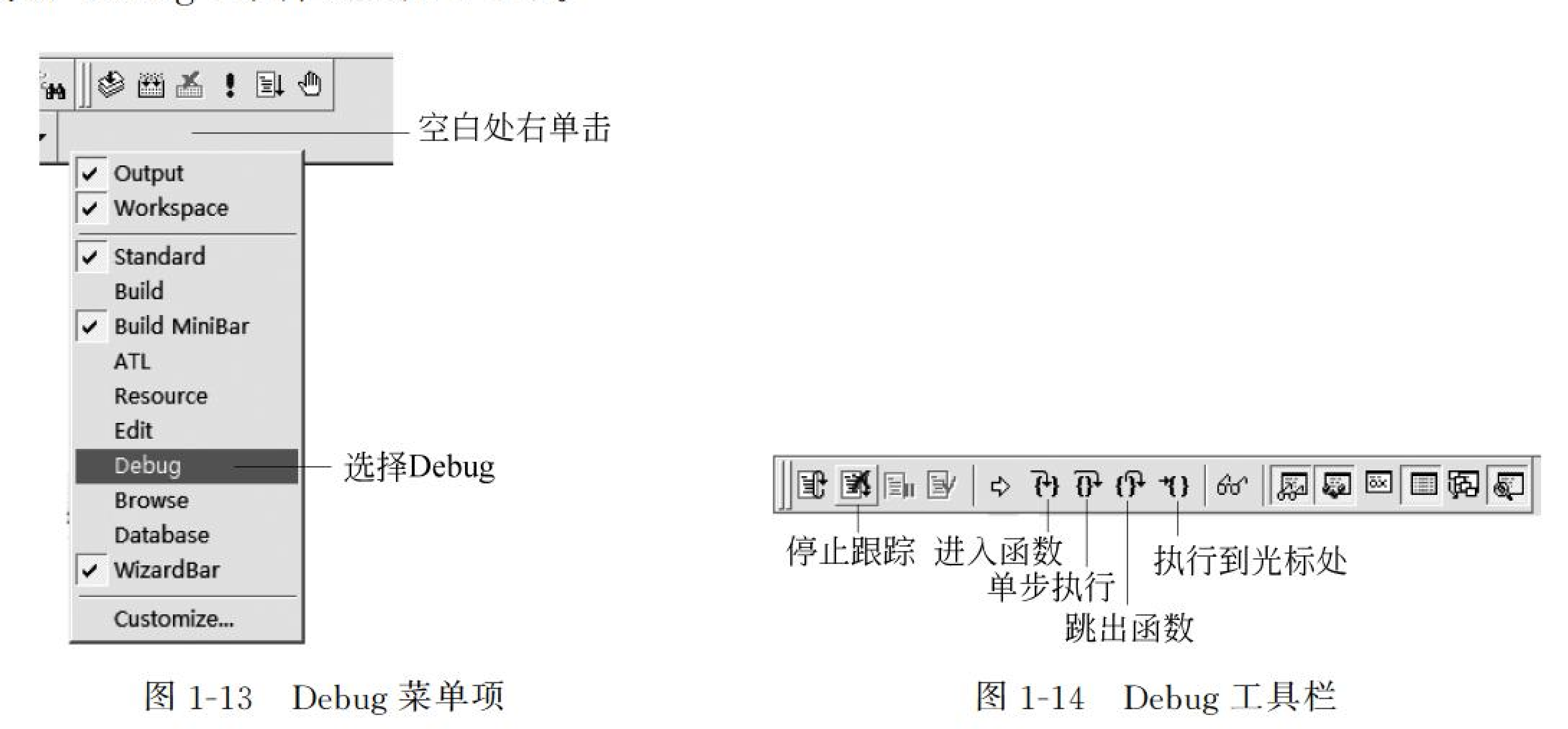



图 1-13 Debug 菜单项　　图 1-14 Debug 工具栏

其中：

Step over（单步执行，或单步跟踪），每单击一次，程序执行一行。

Step into（进入函数），遇到函数时，进入函数，从函数的第 1 个语句开始执行。

Step out（跳出函数），跳出当前函数体，回到主调函数的调用处。

Stop Debugging（停止跟踪），停止跟踪，回到编辑状态。

单步跟踪调试一般是使用 Step over 按钮，一步一步执行；需要时使用 Step Into，进入函数后再使用 Step Over 单步执行。每执行一步，需要时，可以在 Watch（监视）窗口（见图 1-15）中的 Name 列输入变量的名字或表达式，在 Value 列查看变量或表达式的值。注意，对于测试的每一个例子，程序执行到每一步，每一个变量应得到的结果都应该是已知的、确定的。这样，可根据逻辑分析的结果和观察到的变量的值进行比较，如果相同，说明程序执行到此是正确的；如果不相同，说明程序执行到此是错误的，错误在该行或在前面。

**2. 观察内存值**

如果需要，在跟踪过程中可以观察内存中个单元的值。方法是：

（1）打开 Memory 窗口。执行 View|DebugWindows|Memory 菜单命令（见图 1-16）。

（2）在 Address（地址）栏中输入内存地址（见图 1-17 左）。其中的地址可以在 Watch 窗口中输入获取地址的表达式来获得（图 1-17 右，&a 就是获得变量 a 的地址）。

**3. 设置断点**

如果程序很长，单步跟踪就要花费很多工夫。如果知道错误的大概位置或范围，可以让程序直接执行到该处，再进行单步跟踪。方法是在大概位置处设置断点。

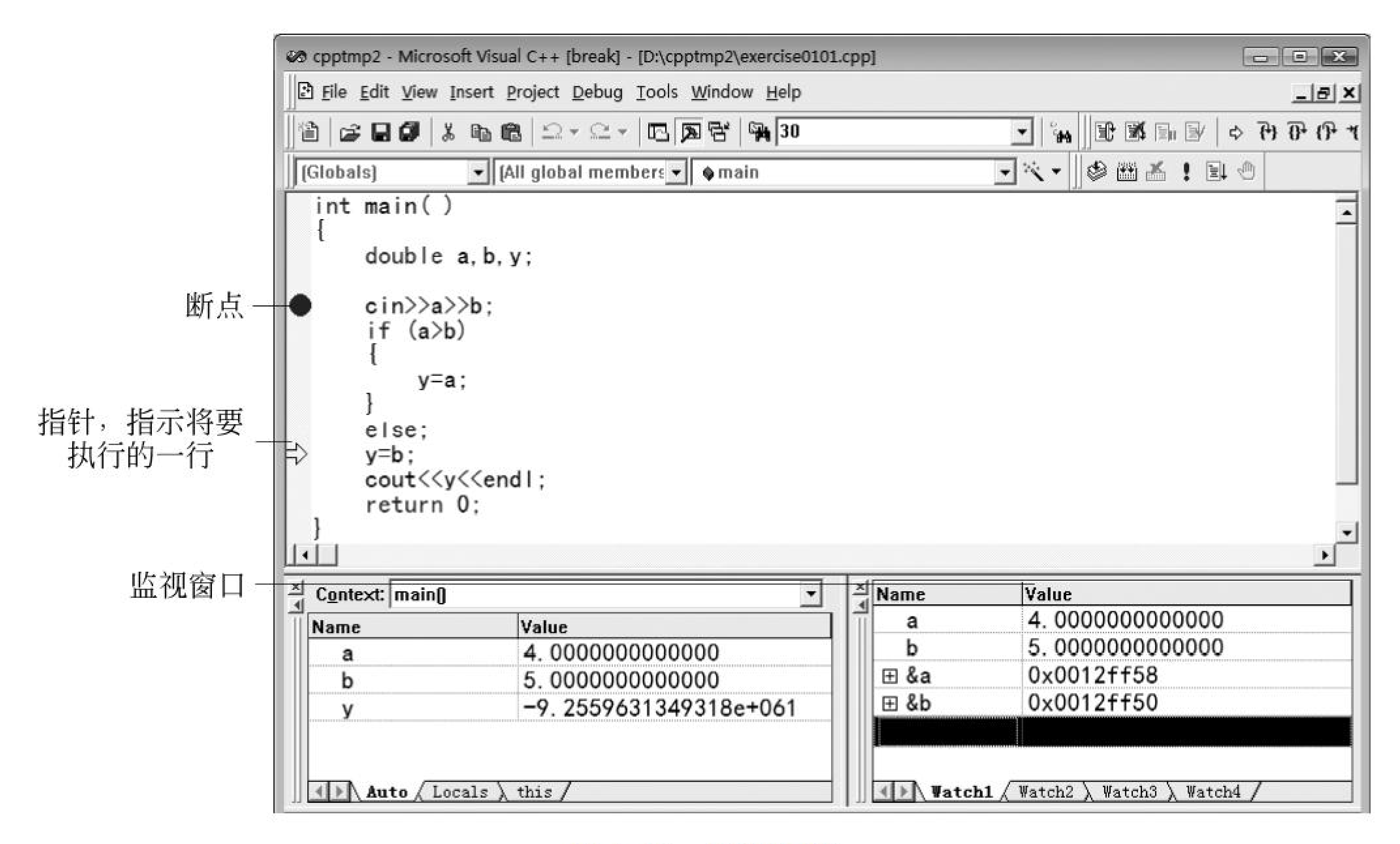



图 1-15　跟踪程序

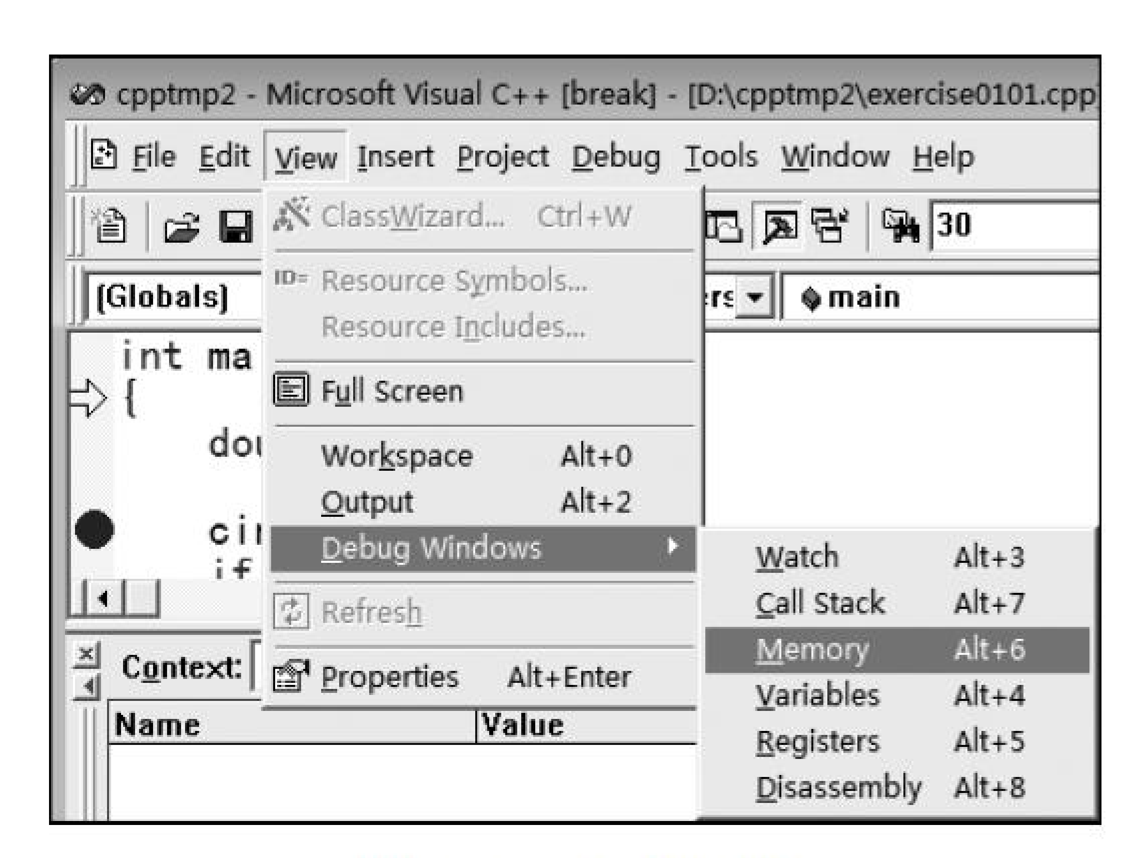



图 1-16　内存菜单

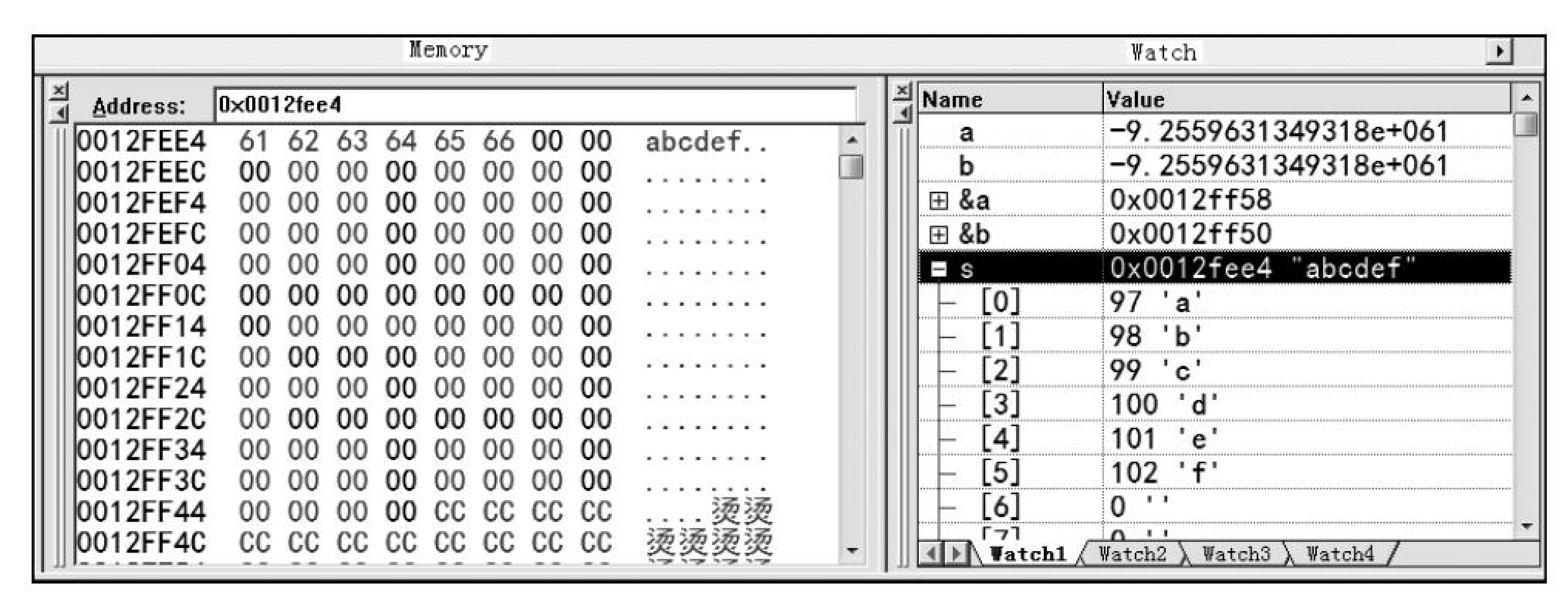



图 1-17　Memory 窗口

(1) 单击工具栏手形按钮 (insert/remove Breakpoint),设置(或取消)断点。

(2) 单击工具栏书形按钮 (go),执行到断点处。

断点可以设置多个,再次单击 go 按钮,执行到下一个断点;再次单击手形按钮,取消断点。

调试程序时,常常是设置断点,执行到断点,根据需要使用"单步跟踪"或"执行到下一个断点"。

### 1.1.6 在一个工程中编辑多个程序文件

一个工程可以有多个 cpp 文件,可以使用 File|New 多次创建,它们之中只允许有一个 main 函数。如果多个文件中有 main 函数,连接时就会出错。对于编写实用的软件,这不是问题。因为一般一个工程是一个软件,只有一个 main 函数,即便有多个 cpp 文件也是如此。

对于 C++ 的学习,一次需要练习编写多个小软件。每个小软件是一个 cpp 文件,各有一个 main()函数。这可以创建多个工程,每个小软件一个工程,这是可以的。但创建多个工程需要花费不少时间,显得很烦琐。实际上,可以在一个工程中编辑多个软件文件。方法如下:

(1) 创建工程。

(2) 编辑第 1 个程序,编译、连接、调试、运行。

(3) 创建第 2 个 C++ source File。

(4) 在工程管理窗口的 FileView 视图中,选择第 1 个文件的文件名,按 Delete 键从工程中剔除它,而不是删除(见图 1-18)。

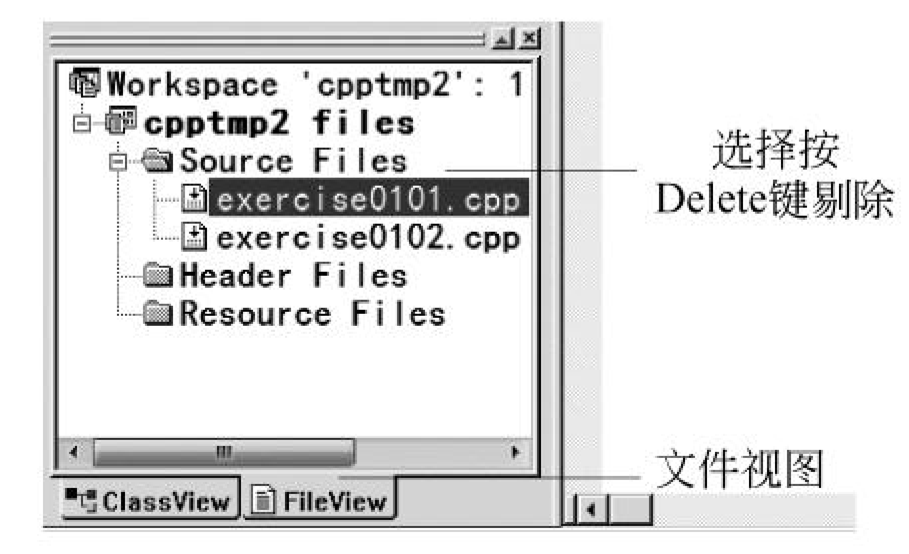



图 1-18 剔除多余的文件

要知道的是,在工程窗口中删除的文件只是说明该文件不再属于这个工程,并没有从磁盘中删除。它仍可以在工程文件夹中找到,可以复制、备份。

### 1.1.7 使用帮助

(1) 使用 Help 菜单。

(2) 在源文件编辑窗口中选择关键词、库函数名、系统的类名的词语,按 F1 键。

## 1.2 Visual C++ 2010 编写控制台应用程序

VS 2005、VS 2008 和 VS 2012 等新版本的用法与 VS 2010(Visual Studio 2010)的用法类似。

### 1.2.1 启动 Visual Studio 2010 集成开发环境

在 Windows 7 系统中,单击"开始"按钮,选择"所有程序"。单击"Microsoft Visual Studio 2010"菜单项,再单击"Microsoft Visual Studio 2010"应用程序选项(如图 1-19 所示),就可启动集成开发环境。在 Windows XP 中的启动方法与此类似,不再赘述。




图 1-19 “新建项目”对话框

## 1.2.2 创建或打开 Win32 控制台工程

### 1. 创建仅含有一个 C++ 文件的工程

在 Visual Studio 2010 中，单击“文件”菜单，选择“新建”，再选择“项目”，就可启动“新建项目”对话框（如图 1-19 所示）。

在新建项目对话框中要做下面设定：

(1) 在左侧窗口中选择编程语言为 Visual C++。这是因为 Visual Studio 2010 将 C++、C#、VB.NET 等多种语言都集成在一个编程环境中，因此需要先选择编程语言。

(2) 在中间的窗口中，选择工程类型为“Win32 控制台应用程序”。

(3) 输入工程名称。

(4) 设定工程的存储目录。

在这些设置完成之后，按“确定”按钮就可进入工程向导对话框，如图 1-20 所示。

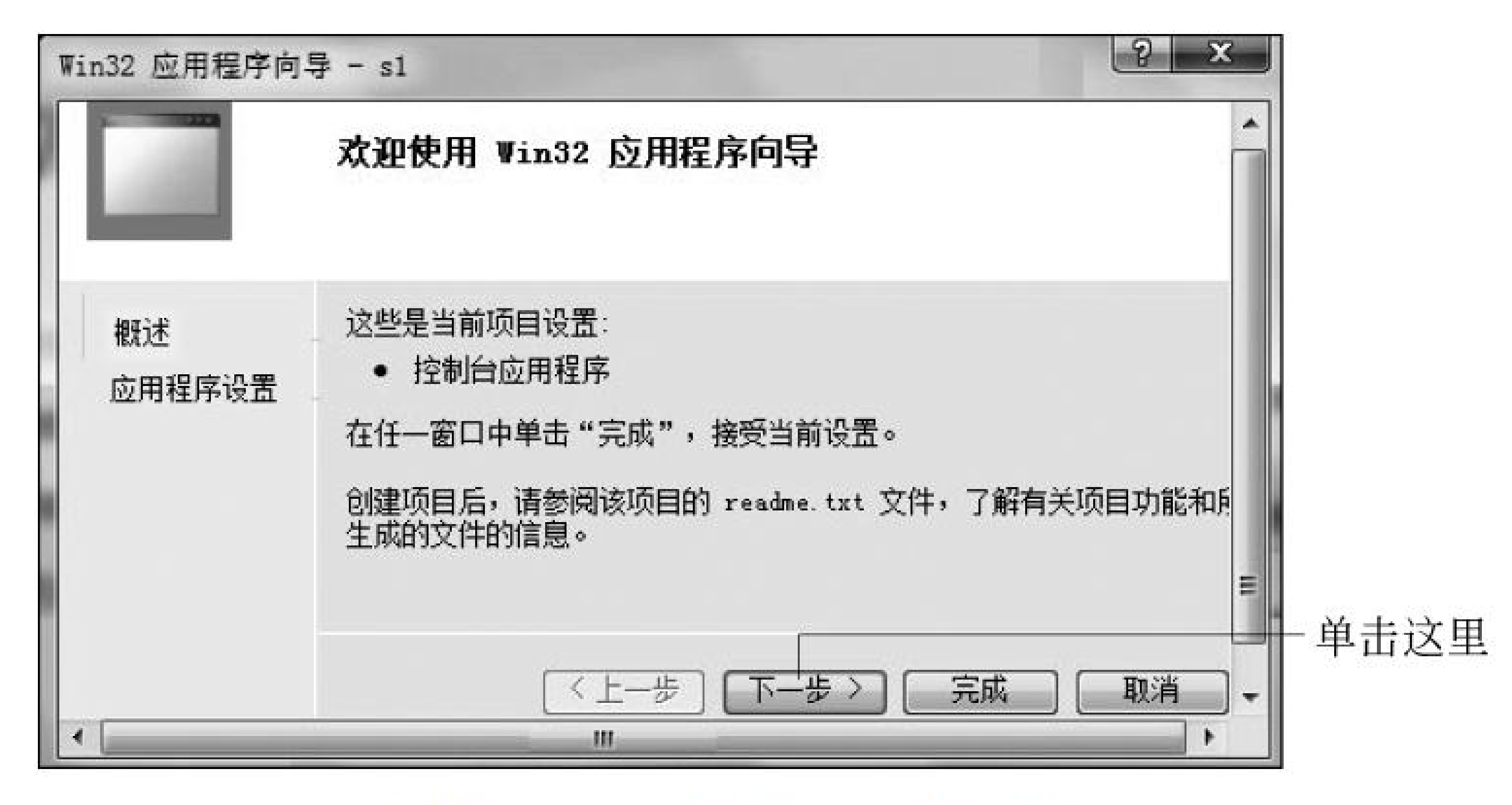



图 1-20 “应用程序向导”对话框

这时候要特别注意，不要在图 1-20 的应用程序向导对话框中按“完成”按钮，那样会

生成一个包含多个预定义文件的工程。在图 1-20 中单击“下一步”按钮，会进入应用程序设置对话框，如图 1-21 所示。在图 1-21 中，设定“附加选项”为“空项目”，再按下“完成”按钮，这样就创建了一个不含有任何文件的空项目。

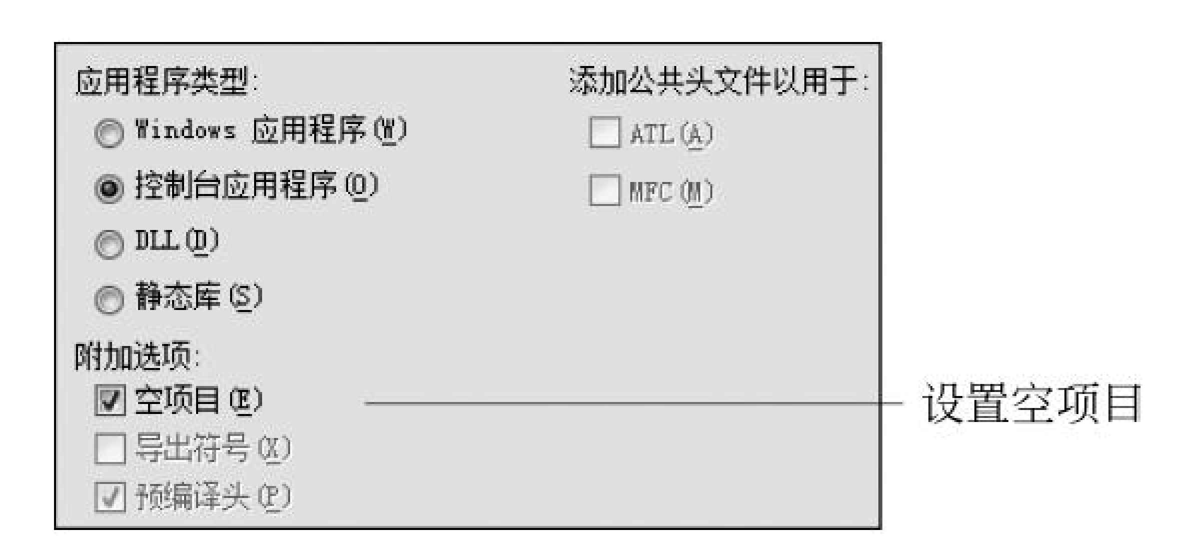



图 1-21　设置应用程序为空项目

下一步要向工程添加一个 C++ 源程序文件。

在“解决方案资源管理器”窗口，用鼠标右键单击“源文件”项目，在弹出菜单中选择“添加”。在子菜单中再次选择“新建项”，就会弹出“添加新项”对话框。这一过程如图 1-22 所示。

在“添加新项”对话框中，选择“C++ 文件(.cpp)”作为添加内容(如图 1-23 所示)，然后输入该文件的名称，再单击“添加”按钮。至此，就创建了一个只包含一个 C++ 文件的 Win32 控制台工程项目。用类似方法可添加头文件，也可添加多个 C++ 源文件或头文件。




图 1-22　在“源文件”中添加新项目

### 2. 打开现有的工程项目

单击“文件”菜单，选择“打开”菜单项；再次选择“项目/解决方案”子项，就调出了“打开项目”对话框。在该对话框中，打开原有工程项目目录，选择工程文件名即可打开。工程文件名的图标如图 1-24 所示，工程文件名的后缀是“.sln”(有可能没显示出来)。




图 1-23　添加 C++ 文件




图 1-24　工程文件名

## 1.2.3 编译、调试及运行程序

Visual Studio 2010 窗口的总体布局如图 1-25 所示。各窗口均可拖动停靠在其他位置。图 1-25 中，“解决方案资源管理器”窗口相当于 VC++ 6.0 的文件视图窗口；而“源文件编辑”窗口、“输出”窗口则和 Visual C++ 6.0 一样。




图 1-25　总体窗口布局

### 1. 编译及连接工程项目

“生成”菜单用于编译及连接工程。它包括编译连接整个解决方案或编译连接单个项目两方面的操作（一个解决方案可以包含多个工程项目），如图 1-26 所示。如果整个解决方案中只有一个项目，那么这两方面的菜单命令效果相同。

注意，图 1-26 中的“编译”命令仅仅用于编译并生成 obj 文件，而形如“生成……”的命令则用于编译并且生成可执行文件。




图 1-26　“生成”菜单

## 2. 调试工程项目

“调试”菜单提供了调试程序的一些操作。它包含了调试状态启动程序和非调试状态启动程序两种方式，如图 1-27 所示。若要调试程序，则必须以调试状态启动程序。

通过图 1-27 中的“启动调试”命令或工具栏中的绿色三角形按钮（见图 1-25）都可以以调试方式启动程序。启动时如果系统发现工程没有编译，则会先编译工程项目。若编译正确并生成了可执行文件，则会进一步进入调试运行状态。

图 1-27　“调试”菜单

1）添加/删除断点

如果程序中没有断点，则程序会不停地按顺序执行下去，因此调试时一般都要设断点。

添加/删除断点的方法主要有以下两种。

方法一：将光标放到想要添加或删除断点的某一行中，再按功能键 F9 添加或删除断点，如图 1-28 所示。

方法二：将鼠标移到某一行前的灰色区域（红色断点所在区域），单击此处可在这一行添加或删除断点。

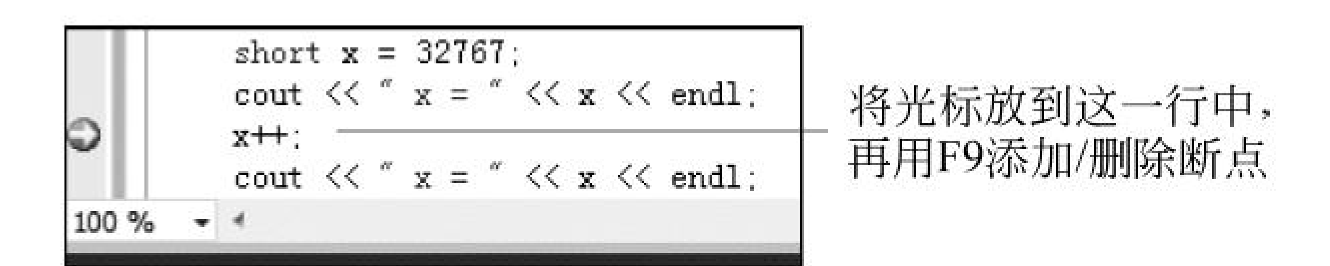



图 1-28　添加/删除断点

2）调试命令及调试工具栏

当按下调试执行的按钮时，在工具栏中会出现如图 1-29 所示的调试工具栏。这个工具栏中的按钮和调试菜单中的命令功能一样。




图 1-29　“调试”工具栏

这些调试命令主要有：启动调试运行、停止调式、重启调试运行、逐句调试（快捷键 F11）、逐过程调试（快捷键 F10）以及跳出当前过程。这些命令的用法与 Visual C++ 6.0 相同。

3）调试中的一些信息窗口

在调试过程中，在编辑窗口下方会出现一些小窗口，它们显示了执行过程中的一些重

要信息。常用的有自动窗口、局部变量窗口、监视窗口、断点窗口等。它们的作用如下。

自动窗口：显示了当前以及前一条语句中的变量名称及其数值。

局部变量窗口：显示了当前函数中局部变量的名称及其数值。

监视窗口：用户可以在此窗口中输入某个变量的名称并监视其变化。

断点窗口：可以管理程序中的断点，比如添加、删除、禁用断点。

## 1.3 C++ Builder 6.0

C++ Builder 6.0 是 Borland 公司推出的基于 C++ 语言的快速应用程序开发(Rapid Application Development)工具。C++ Builder 结合了基于组件的程序设计技术，可视化组件库和编译器、调试器，功能强大。C++ Builder 具有数据库应用程序开发功能，提供众多的数据库感知控件和底层的 BDE 数据库引擎，具有网络编程能力，提供许多 Internet 应用程序开发控件，如 WebBroker、CppWebBrowser 和 WinSocks 等，基本涵盖了 Internet 应用的全部功能，程序员可以方便地建立自己的 Internet 应用程序。其软件界面如图 1-30 所示。

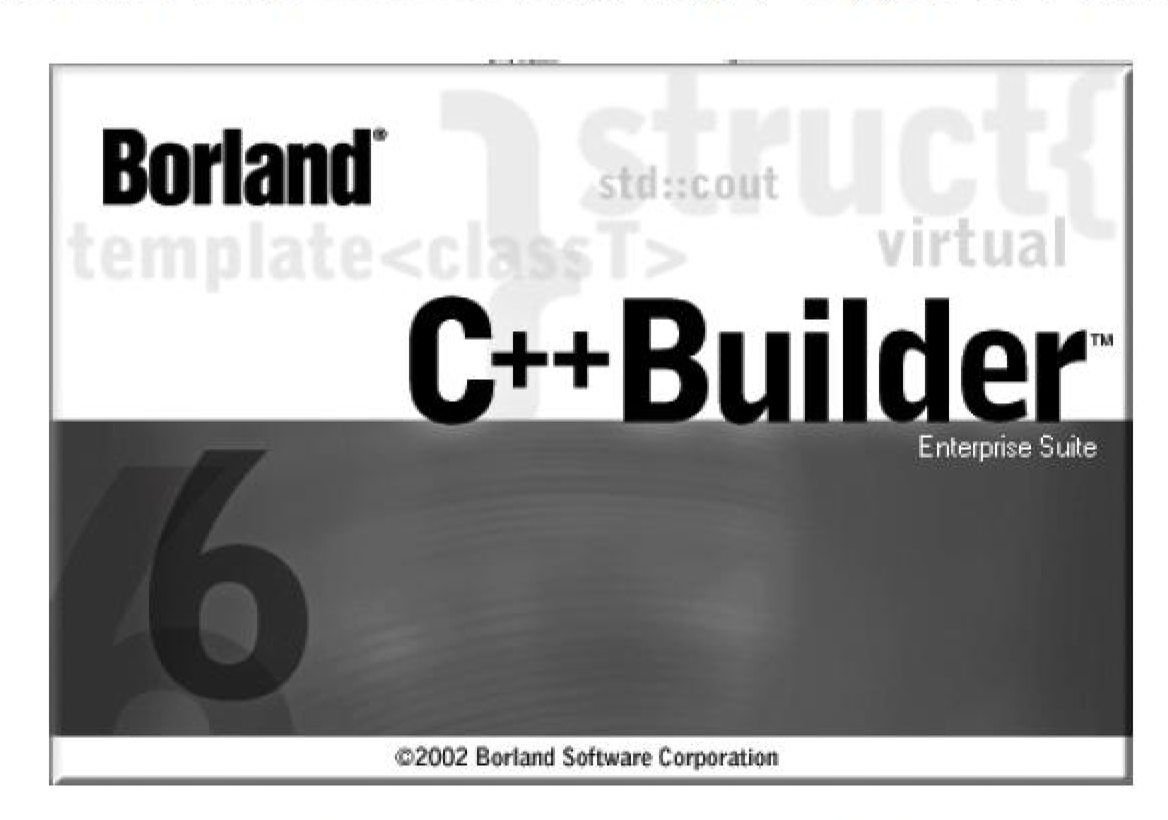



图 1-30 C++ Builder 6.0 软件界面

### 1.3.1 下载与安装

到网站 http://www.borland.com/下载“C++ Builder 6.0”软件或更新的版本，双击 INSTALL.EXE 文件可进行安装。

### 1.3.2 基本使用

在 Windows 的开始菜单选择 Borland C++ Builder 6.0|C++ Builder 6.0，并单击，即可打开 C++ Builder 6.0 开发工具的主界面。下面以一个简单的基于字符控制台的“Hello World!”程序为例进行介绍。

#### 1. 创建工程编写程序

(1) 在 C++ Builder 6.0 中选择 File|new|Other 菜单，出现 New Items 窗口，如图 1-31 所示。

(2) 在 New Items 窗口中选择 Console Wizard(控制台程序生成器),然后单击“OK”按钮,出现 Console Wizard 窗口,如图 1-32 所示。

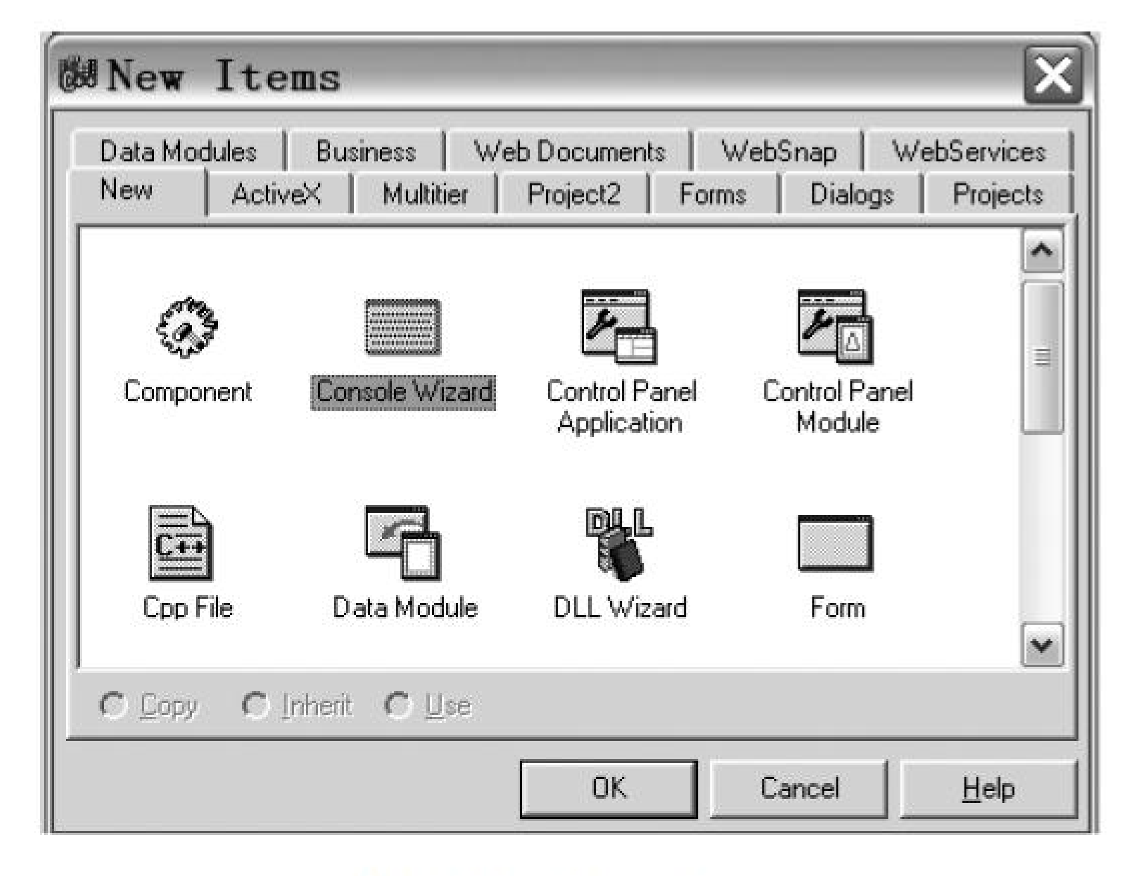



图 1-31　New Items




图 1-32　Console Wizard

(3) 选择 C++ 和 Console Application,然后单击“OK”按钮,并在出现的源程序编辑窗口中输入以下代码(如图 1-33 所示):

```
#include <iostream>
using namespace std;
int main()
{
    char word[]="Hello World!";
    cout<<word<<endl;
    cin>>word;
    return 1;
}
```

单击 Project|Compile Unit(编译)菜单,编译成功后,单击 Run|Run(运行)菜单运行,结果见图 1-34。

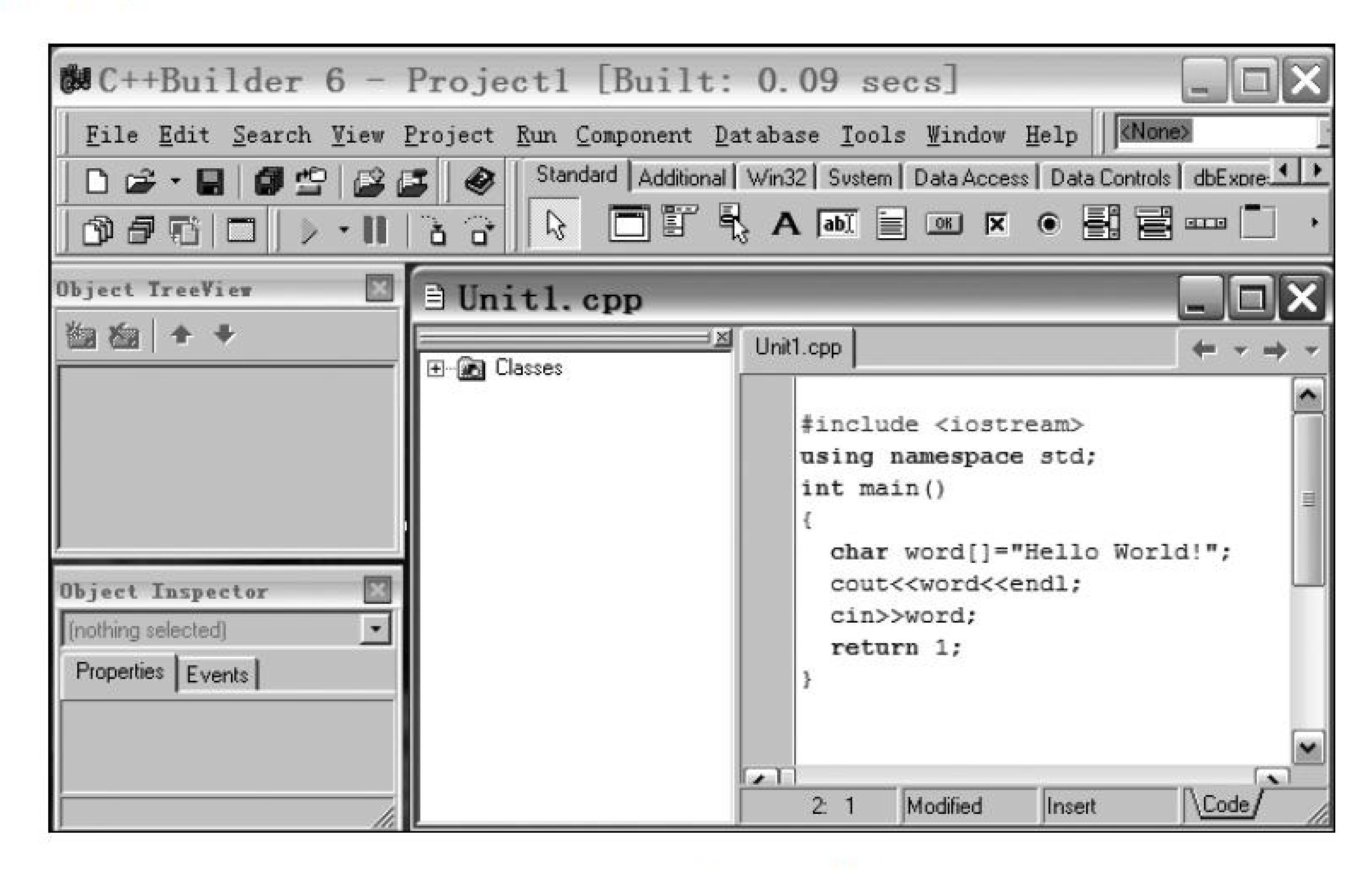



图 1-33　输入代码

**2. 调试**

在源程序的某一行左侧单击，设置调试断点，然后重新运行程序，进入调试运行环境。此时可以选择 View|Debug Windows|Watches，在出现的观察变量窗口内，右击 Add Watch，在出现的 Watch Properties 窗口中输入 expression 的项，比如 word。此时可以观察到 word 变量的值，如图 1-35 所示。

单击 Run|Program Reset 菜单，可以退出调试运行环境。

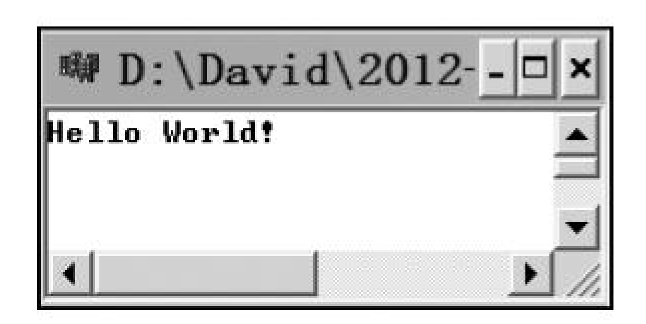



图 1-34　运行结果

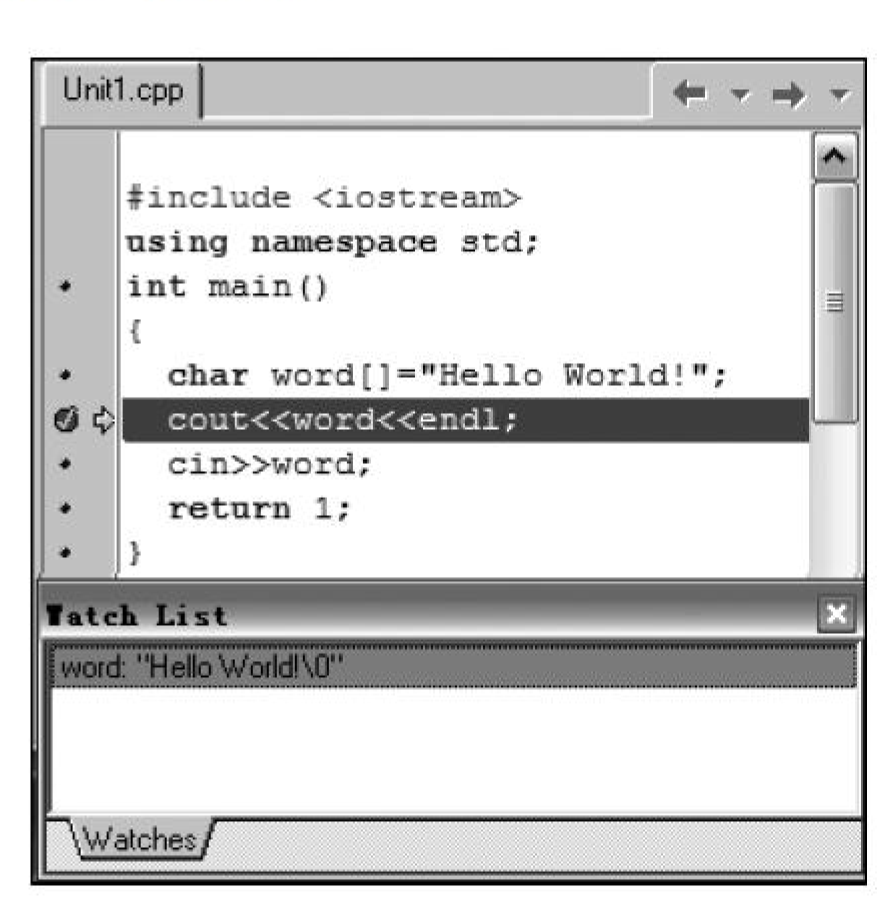



图 1-35　Eclipse 中的调试

# 第 2 部分

## 实 验 指 导



## 2.1 实验 1 显示程序和简单计算程序

**实验目的**

(1) 掌握 C++ 程序的基本结构。

(2) 掌握一种集成开发环境的基本使用方法。

(3) 掌握工程、文件的创建方法,掌握 C++ 程序的编译、链接和运行方法。

(4) 掌握 C++ 中的输出、输入语句,算术运算。

### 2.1.1 显示由"*"组成的矩形

在屏幕上实现由*组成的宽度为 30、高度为 10 的矩形。

**【本例目的】** 练习编程环境的使用,了解 C++ 程序的基本结构,学会简单信息的表示和输出。

**【问题分析】** 本例实现的矩形如下(见图 2-1):

```
******************************
*                            *
*                            *
*                            *
*                            *
*                            *
*                            *
*                            *
*                            *
******************************
```

图 2-1 矩形

其中,第 1 行和最后一行各由 30 个星号组成,中间 8 行,两头是*号,中间是空格。

**【算法描述】**

(1) 显示 30 个星号,换行。

(2) 显示"*                            *",其中两个*之间有 28 个空格。

(3) 重复②7 次(连同原来的②,共显示"*    ... *"8 行)。

(4) 再显示 30 个星号,换行。

**【编程指导】**

(1) 创建工程。请在 D 盘或 E 盘中创建工程,工程名建议为 lab01。注意不要使用默认路径,工程名要有意义。

(2) 创建 C++ 源程序文件。建议文件名为 lab0101.cpp(其中的扩展名是系统自动添加的)。

(3) 编写程序。本例组成矩形的符号、宽度、高度都是固定的,所以只要使用 cout 语句完成即可。

(4) 编译、连接。

(5) 运行。

(6) 如果有错误,则检查程序,修改错误,再重新编译、连接、运行,直到结果正确。

**【注意事项】** 使用 cout 时,可以使用一个 cout 输出多项数据,每个数据前面都有一对"<"符号,例如:

```
char name[20]="david";
cout<<"My name is "<<name<<endl;
```

上面的 cout 语句中,输出了三项内容。第 1 项是"My name is ",这是一串固定的字符,称为**字符串常量**;第 2 项是 name,它的值是"david";第 3 项是 endl,这是一个控制符号,输出一个 endl,就是输出一个**换行符**,那么以后输出的内容会显示在下一行。可以去掉最后的"<<endl"或多写几个试试,比如 cout<<endl<<endl<<endl;。

**【问题扩展】** 本例显示的是一个固定的矩形,每次运行都一样。不过随着新知识和技能的不断学习,可以实现让用户输入组成矩形的符号。比如用#、%等组成,宽度和高度也可以让用户选择。还有,高度为 10 时,本例用了 8 个语句:

```
cout<<"*                                    *"<<endl;
```

如果高度为 100 呢?学了第 3 章的循环控制结构,即使是 1000 行,也很容易实现。例如将程序中的显示矩形中间边框的 8 行语句替换为:

```
int N=30;
for(i=0;i<N;i++)      //N的值就是行数,前面 N=30,就是显示 30行,改成 20就显示 20行
{
    cout<<"*                                    *"<<endl;
}
```

试试?

## 2.1.2 计算立方体的周长、表面积和体积

用户输入立方体的长、宽和高,计算立方体的周长、表面积和体积并显示输出。

**【本题目的】** 练习简单的计算,运算符、表达式、赋值、输入、输出。

**【问题分析】** 立方体的周长是:(长+宽+高)×4。

表面积是:(长×宽)×2+(长×高)×2+(宽×高)×2。

立方体的体积是:长×宽×高。

因此,知道了长、宽、高,这三个量是容易计算的。

**【算法描述】**

(1) 输入立方体的长、宽、高。

(2) 依次计算立方体的周长、表面积和体积。

(3) 显示立方体的周长、表面积和体积。

**【编程指导】** 编程首先解决数据的表示问题,然后再说输入、计算和输出。本例长、宽、高和周长、表面积以及体积分别用一个符号表示。程序设计中表示数据的符号可以用

一串字符表示，如，可以使用 length、width、high、circumference、surface_area 和 volume 分别表示上面的量，而且在使用前要先声明。如果允许长、宽、高为实数，上面的符号的声明语句为：

```
double length, width, high, circumference, surface_area, volumn;
```

输入使用 cin 语句。一个 cin 可以有多个输入项，每个输入项前要有一对>(大于)符号。例如输入长、宽、高的语句可以写为：

```
cin>>length>>width>>high;
```

计算的问题比较简单，只需要写出运算的表达式即可。如：

```
circumference=(length+width+high)*4.0;          //计算周长
```

输出使用 cout，参考题目 1，把字符串常量改为本例中的 circumference、surface_area 和 volume 三个变量。

**【测试指南】** 为了检查程序中是否存在错误，运行程序，输入三个数如 2 3 4(中间用空格隔开)。先用手工计算周长、表面积和体积，再与程序的运行结果比较。由于允许输入为实数，再输入 2.1,3.2,4.3 三个数，看结果是否正确。输入负数呢？

**【注意事项】** double 说明后面的符号用来表示双精度实数。双精度实数在计算机中常用 8 个字节表示，与此相对的是单精度实数，用 float 说明，用 4 个字节表示，所以双精度的实数可以有更高的精度，能表示的数的范围更大。用 int 说明表示整数的符号(这些符号称为变量)。

另外要注意，语句的执行是有顺序的。C++ 程序从 main 函数开始，从上到下顺序执行语句，所以，赋值语句和输入语句一定在计算语句之前，才能按已赋的值进行计算，否则得不到正确的结果。例如：

```
int a,b,c;
c=a+b;
cin>>a>>b;
cout<<c<<endl;
```

运行此程序，观察运行结果，分析原因并改正。

**【问题扩展】**

(1) 学会了本例，一般的计算程序都可以按照图 2-2 的步骤编写。

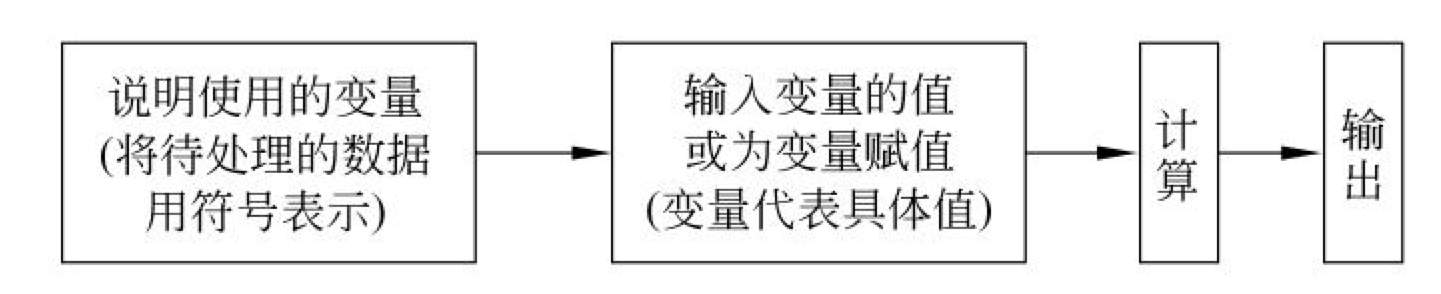



图 2-2　编程的步骤

C++ 中表示算术运算加、减、乘、除的符号是＋、－、*、/，% 表示求余数，例如 a＝75%8，得到的结果是 3。C++ 中没有乘方运算符号，计算乘方使用连乘的方法。例如 a＝

x * x * x,得到 x 的立方。C++ 中可以使用小括号表示优先进行的运算。例如 y=(a+b)*(c+d),计算机会先计算 a+b,再计算 c+d,然后再把两个结果相乘的结果赋值给 y。

(2) 编程计算万有引力

$$F = G\frac{m_1 \times m_2}{R^2}$$

其中 G 是万有引力衡量,$G=6.67\times10^{-11}N\cdot m^2/kg^2$,$m_1$、$m_2$ 是两星球的质量,R 是两星球的距离,其中力的单位是牛顿(N),距离的单位是米(m),质量的单位是千克(kg)。其中的质量和距离由用户输入。

**提示**:C++ 中,表示 10 的多少次方使用 E±N(N 是常数),例如,double G=6.67E-11 就是上述万有引力恒量的表示方法。

## 2.1.3 计算简单数学函数的值

对数螺线是自然界中许多事物的轮廓线,对数螺线在极坐标系中的解析式为

$$\rho = \alpha e^{\beta\theta} \tag{1}$$

其中 $\rho$ 是极径,$\theta$ 是极角,单位为弧度,$\alpha$、$\beta$ 是参数。极坐标和直角坐标系的变换为:

$$x = \rho \times \cos\theta,\quad y = \rho \times \sin\theta \tag{2}$$

编写程序,用户输入一个以度为单位的角度(极角),计算对数螺线上的直角坐标系中的坐标,其中 $\alpha$ 取 1.0,$\beta$ 取 0.1。

**【本例目的】** 练习算术运算符、表达式和数学函数的使用。

**【问题分析】** 就解题来说,暂时不考虑对数螺线是什么。其实就是给出以度为单位的角度,先转换为弧度,再用式(1)计算 $\rho$,然后再用式(2)计算 x、y,最后显示即可。现在的问题是函数如何计算?

对于常用函数的计算,C++ 编译器中已经内置了计算这些函数的程序,只要学会使用即可,当然以后需要时也可以编写计算自定义的函数的值。使用常用的数学函数,需要在 main 函数前包含头文件 cmath,即写上如下语句:

```
#include  <cmath>
```

若 theta 是以弧度为单位的角度,则

```
y=exp(theta);
x1=cos(theta);
y1=sin(theta);
```

分别得到 $e^{theta}$、theta 的余弦、正弦的函数值并赋值给相应的变量。exp、sin、cos 是 C++ 中计算 e 的指数、正弦、余弦函数值的函数的名字(称为函数名),括号中是函数的参数。给定参数值,就能把函数值计算出来。

**【算法描述】**

(1) 取 $\alpha=1.0$,$\beta=0.1$,$\pi=3.1415926$。

(2) 输入以度为单位的角度值 theta。

(3) 转换为弧度:theta=theta/180 * $\pi$。

(4) 使用式(1)计算 $\rho$。

(5) 使用式(2)计算 x,y。

(6) 显示 x、y。

**【编程指导】** 程序中没有 $\alpha$、$\beta$、$\pi$ 这样的符号,所以可以给它们取由字母、数字组成的名字,如 alpha、beta 和 pai;并且它们要先声明,后使用。由于是实数,应声明为 double 类型。

另外注意包含头文件 cmath,并注意语句的顺序。

**【测试指南】** 本例设计的输入是一个角度。角度不同,坐标点就会落在直角坐标系的不同象限中,所以,应选不同象限中、不同坐标轴上的角度,看结果是否正确。以下是输入和输出的对应结果:

| 输入(角度) | 输出(x,y) | | 备注 |
|---|---|---|---|
| 0 | 1 | 0 | x 正半轴 |
| 30 | 0.912579 | 0.526877 | 第 1 象限 |
| 90 | 3.13524e−008 | 1.17009 | y 正半轴 |
| 145 | −1.05505 | 0.738753 | 第 2 象限 |
| 180 | −1.36911 | 7.33702e−008 | x 负半轴 |
| 225 | −1.04721 | −1.04721 | 第 3 象限 |
| 270 | −1.28774e−007 | −1.60198 | y 负半轴 |
| 315 | 1.22532 | −1.22532 | 第 4 象限 |
| 360 | 1.87446 | −2.00903e−007 | x 正半轴,曲线绕过一周 |

**【问题扩展】** 袅袅上升的炊烟、轻轻荡开的涟漪、缓缓攀援的蜗牛、携拥旋舞的繁星,它们的形状都可以用螺线描述。

对数螺线是由笛卡儿(René Descartes)在 1638 年发现的。雅各布·伯努利(Jakob Bernoulli)后来重新研究之。他发现了对数螺线的许多特性,如对数螺线经过各种适当的变换之后仍是对数螺线。他十分惊叹和欣赏这曲线的特性,故要求死后将其刻在自己的墓碑上,并附词“纵使改变,依然故我”(eadem mutata resurgo)。可惜雕刻师误将阿基米德螺线刻了上去。

e 在数学上是 $\left(1+\frac{1}{x}\right)^x$ 当 x 趋于无穷时的极限,称为自然对数的底。自然界中许多现象如物体的冷却、细胞的繁殖、放射性元素的衰变、宇宙的形成、生命的进化等都与 e 有关。人们常常把 e 或由 e 经过一定变换和复合的形式定义为“自然律”。

(1) 修改程序,让用户输入 $\alpha$、$\beta$ 和角度,计算 x、y。

(2) 修改程序,让用户输入 $\alpha$、$\beta$,自动计算极角在 $0\sim4\pi$ 内的 100 个对数螺线的直角坐标系坐标点。

(3) 用户输入一个以度表示的角度,利用 C++ 的库函数,计算其正弦、余弦、正切和余切的函数值。

## 2.1.4 按方阵格式显示数据

用户输入 5 行数据,每行中的数据间用空格隔开。数据如下:

| | | | | |
|---|---|---|---|---|
| 1 | 美国 | 46 | 29 | 29 |
| 2 | 中国 | 38 | 27 | 23 |
| 3 | 英国 | 29 | 17 | 19 |
| 4 | 俄罗斯 | 24 | 26 | 32 |
| 5 | 韩国 | 13 | 8 | 7 |

请将输入的数据以下列形式显示出来：

2012年伦敦奥运会奖牌榜

```
**********************************
序号  参赛队  金牌  银牌  铜牌  合计
 1    美国    46    29    29    104
 2    中国    38    27    23    88
 3    英国    29    17    19    65
 4    俄罗斯  24    26    32    82
 5    韩国    13    8     7     28
**********************************
```

注意，其中的列要对齐。

**【本题目的】** 练习字符串、整数的输入、输出，转义字符的使用，格式对齐。

**【问题分析】** 本例实际是练习数据的表示、数据的输入和输出的格式。本例有5行数据，各行的格式是一样的。每行有5个整数、一个字符串(若干个字符)。整数用int说明，例如int a；一次可以说明多个变量，中间用逗号隔开，例如int a,b,c；输入用cin，如cin>>a；一个cin可以输入多个数据，中间加多个>>号，例如cin>>a>>b>>c；字符串用char说明，如char s[20]；说明s可以表示一串字符，长度不超过20－1＝19。本例的输出是关键。如果输出的数据之间用空格隔开，那么由于数据的长度不同，各列数据很难对齐。C++中，列对齐的方法之一是使用'\t'。使用'\t'可以控制后面的输出在下一个制表位上。制表位约定了字符显示在哪一列上。第1个制表位是第9列，第2个制表位是第17列，第3个制表位是第25列，依次类推。两个制表位之间相差8个字符。

本例的一组数据的输入和输出可以编程如下：

```
int a1,a2,a3,a4,a5;
char name1[30];
cin>>a1>>name1>>a2>>a3>>A4;
a5=a2+a3+a4;
cout<<a1<<'\t'<<name1<<'\t'<<a2<<'\t'<<a3<<'\t'<<A4<<'\t'<<A5<<end;
```

**【编程指导】** 本例需要输入输出5行数据。可以写5行输入和5行输出，也可以尝试使用循环。

**【问题扩展】**

(1) ASCII字符集中，字母、数字、标点符号是可显示的字符。还有一些符号起控制作用，是不可显示的字符，如回车、换行、响铃等。要使用不可显示的字符，可以使用转义符号。如'\t'表示制表符，'\n'是换行符等。常用转义符见表2-1。

表 2-1　常用转义符

| 转　义　符 | 含　　义 | 转　义　符 | 含　　义 |
|---|---|---|---|
| \a | 响铃 | \r | 回车 |
| \n | 换行 | \\ | 字符\ |
| \t | 水平制表符 | \" | 双引号 |
| \b | 退格 | \' | 单引号 |

请同学们在输出的字符串中加入表 2-1 中的转义符，看显示效果如何？一次使用一个。

（2）进行格式控制的另一种方法是使用流操纵符。例如，cout<<setw(n)<<a;输出的数据 a 将占用 n 个字符的位置。例如：

```
int a1=11,b1=12345;
int a2=12345,b2=53;
cout<<setw(10)<<a1<<setw(10)<<b1<<endl;
cout<<setw(10)<<a2<<setw(10)<<b2<<endl;
cout<<"12345678901234567890"<<endl;
```

显示效果为：

```
        11     12345
     12345        53
12345678901234567890
```

**注意：**

① 使用此方法，要在"#include<iostream>"的下面添加一行程序：

```
#include <iomanip>
```

② 输出第 3 行是为了看到上面两行输出的每一个数据占据的字符宽度。

# 2.2　实验 2　简单信息的表示和数据计算

**实验目的**

（1）理解常量、变量、数据类型等基本类型。
（2）理解变量的声明。
（3）掌握常量的声明方法，掌握变量的声明、初始化、赋值等方法。
（4）掌握运算符和表达式。
（5）掌握运算符的顺序，不同类型数据的混合运算和转换方式，强制类型转换。

## 2.2.1　数学函数计算

编程试求函数

$$y = \frac{\sin x^2}{1 - \cos x}$$

当 x→0 时的极限。提示：三角函数的值是通过数学函数 sin(x)(正弦)、cos(x)(余弦)来计算的(函数使用见附录)。输入的数值逐步变小，不要输入 0。

**【本题目的】** 练习运算符、表达式、数学函数、乘方、赋值，用计算机探究数学问题。

**【问题分析】** 本题的关键是数学函数的使用：

① 包含头文件<cmath>：#include <cmath>；

② 函数的调用 y=sin(x)，x 是自变量，单位为弧度，结果赋值给 y；

③ 算式分子中是 x 的平方。

**【算法描述】**

(1) 输入角度；

(2) 转换为弧度；

(3) 计算 y；

(4) 输出 y。

**【测试指南】** 测试的数据可以从 1 开始，每次除以 2 或除以 3，逐渐减小；将输入输出结果列在一个表格中，观察结果趋势。

**【问题扩展】** 试求下式的极限。

$$\lim_{x \to 0} \frac{(1 + 2x)^{\sin x} - \cos x}{x^2}$$

**提示**：指数函数用 pow(x,y)，计算 x 的 y 次方，但对 $x^2$ 请使用 x * x。

## 2.2.2 信息加密

输入一个小写字母，显示其 ASCII 码；将其转换为其后的第 3 个大写字母，显示转换后的字母及 ASCII 码('x','y','z',其后的第 3 个大写字母分别为'A','B','C')。输入输出格式如下所示：

```
a
原字母:a  ASCII: 97
加密后:D  ASCII: 68
```

其中 ASCII 值显示的是十进制格式。

**【本题目的】** 掌握字符的概念、字符的表示、ASCII 码、字符的运算、字母的大小写转换。

**【问题分析】** 本例有两个关键问题，一是加密，二是显示 ASCII 码。在 C++ 中，字符实际存放的是其 ASCII 值，也是一个整数。由于在 ASCII 表中，字符的编码是连续的，所以一个字符后的第 3 个字符就是将该字符的 ASCII 值加 3. 然而对于字母 x,y,z,加 3 后，就不再是字母的 ASCII 值了，为了能够循环(x,y,z,a,b,c,…，即数到 z 后再从 a 开始数 1,2,3)可以做如下计算。设 a 表示输入的字符，b 表示加密后的字符：

① b=a-'a';  //如果 a 是小写字母，则其值在 97～122 之间，'a'的值为 97，则 b 的值是一个 0～25 的整数，是 a 在字母表中的序号(从 0 开始)。

② b＝b＋3；　//数值加 3，是 a 后第 3 个字符在字母表中的序号。a 是'x'，'y'，'z'时，这个序号是 26，27，28，但它们实际不对应字母的序号了。

③ b＝b％26；　//％是求余运算，即 b 除以 26 得到的余数，取值 0～25。这样当 b 的值为 26，27 和 28 时，结果分别会是 0，1，2，这对应字母'a'，'b'，'c'在字母表中的序号，刚好满足本题的加密要求。

上述三步使得 b 变成 a 后面的第 3 个字符在字母表中的序号，且 a 为'x'，'y'，'z'时，b 为'a'，'b'，'c'的序号。

④ b＝b＋'A'；　//右边 b 值在 0～25 之间，'A'的值为 65，则右边表达式的计算结果在 65～90 之间，是大写字母的 ASCII 值的区间。

这就将字符 a 转换为其后第 3 个大写字符(其后第 3 个字母的大写形式)。

至于显示 ASCII 值，就看怎样看待整数值了。设 a 为字符型变量，在内存中存放的是一个整数，其值是字符的 ASCII 值，a 是字符型，cout<<a；显示的是某个字符；而(int)a 可以将 a 强制转换为整型，cout<<(int)a；显示的就是 a 代表的字符的十进制 ASCII 值。

**【算法描述】**

① 输入一个字符。

② 按问题分析中的步骤进行加密计算。

③ 按要求进行输出。

**【测试指南】** 本例要求将小写字母转换为其后的第 3 个大写字母。输入'x'，'y'，'z'时要循环数。所以，要能处理'x'，'y'，'z'和非'x'，'y'，'z'的情况，所以，请分别输入'a'，'x'，'y'，'z'进行测试。不仅如此，好的程序不仅输入正确时能得到正确结果，输入错误时要能做出判断并给出提示信息。请输入一个大写字母，输入一个数字，输入一个标点符号或汉字试试，程序如何？不过，本实验暂时不解决这一问题，等学过分支语句后，这个程序就可以编写得更好了。

**【问题扩展】** 请试着编写求解下列问题的程序。

① 将小写字母转换为其后的第 k 个大写字母，k 和待转换的字母由用户输入。

② 将大写字母转换为其前的第 k 个小写字母，k 和待转换的字母由用户输入。

③ 输入一个 5 字母的小写英文单词，输入密钥 k，将单词中的每个小写字母用其后的第 k 个大写字母替代，输出替代后的单词。

④ 编写③的解密程序。

⑤ 通过异或运算加密解密。输入一个四字母的单词为原文，再输入一个整数作为密钥，将原文的每一个字与密钥作异或运算，输出加密后的字符串。

同时这也是一个解密程序。输入密文和密钥，可以得到原文。

## 2.2.3　贪心算法找零钱

为顾客找零钱时，希望选用的纸币张数最少(分，四舍五入)。例如 73.85 元，希望零钱的面值为 50 元 1 张、20 元 1 张、1 元 3 张、5 角一张、1 角 4 张。设零钱面值有 50 元、20 元、10 元、5 元、1 元、5 角、1 角，请编写程序，用户输入 100 以下最多两位小数的实数，计

算找给顾客的各面值的纸币张数,并在程序中想一个验证结果是否正确的办法。

**【本题目的】** 了解贪心策略,探索基本问题的求解方法,求余运算,取整运算,程序验证。

**【问题分析】** 该问题的一般策略是先看应找多少,然后选择能找的最大面值的纸币及张数,从应找中减去已找;再看能找的最大纸币及张数……直到找完。这样的策略,每次都选择当前最优的选择,称为**贪心算法**。这是一种求解问题的策略,在活动安排、最优装载、Huffman 编码中经常使用。

对于本问题,看应找中有多少 50,就是 50 面值的纸币张数。减去后,再看有多少 20,就是 20 元面值的纸币张数……"多少个"相除看商的整数部分,取整即可。

**【算法描述】** 设零钱数量用 cash 表示。

① cash=cash * 10+0.5,单位转换为角;

② change50=[cash/500],cash=cash-change50 * 500,方括号表示取整;

③ change20=[cash/200],cash=cash-change20 * 200;

④ change10=[cash/100],cash=cash-change10 * 100;

⑤ ...

⑥ change01=[cash];

⑦ 显示结果。

**【结果示例】**

```
请输入应找钱数(单位:元)
77.43
50元:1张
20元:1张
10元:0张
5元:1张
1元:2张
5角:0张
1角:4张
```

**【编程提示】** C++ 有取整函数,但 C++ 的整数运算有一条规则是:整数的运算结果为整数,实数赋值给整型变量结果为整数,而且是下取整。例如 1/2 的结果为 0,int a=3.14;结果 a 的值为 3。实数与整数的运算结果为实数,如 1.0/2 结果为 0.5。

**【测试指南】** 本例测试需要一些技巧。应选择应找钱数,使每种面值的纸币都至少出现一次,四舍的要有,五入的要有,多位小数,只有整数的、负数的等。

**【思维扩展】** 贪心算法(Greedy algorithm),又称贪婪算法,是一种在每一步选择中都采取在当前状态下最好或最优(即最有利)的选择,从而希望导致结果是最好或最优的算法。贪心法可以解决一些最优化问题,如:求图中的最小生成树、求哈夫曼编码等。对于其他问题,贪心法不一定能得到所要求的答案。一旦一个问题可以通过贪心法来解决,那么贪心法一般是解决这个问题的最好办法。由于贪心法的高效性以及其所求得的答案比较接近最优结果,贪心法也可以用作辅助算法或者直接解决一些要求结果不特别精确

的问题。

### 2.2.4 整数的分离

在 IC 卡中,存放有 4 个 8 位的二进制数(共 4 字节,32 位),由于系统设计的原因,它们只能一起读写。现将其读取到一个无符号整型变量中,请编写程序,将它们进行分离,即显示出原来的 4 个整数(十进制)。卡中的 4 个 8 位二进制数通过输入的方式进行模拟。

**【本题目的】** 理解数据类型、整数存储、位运算。

**【问题分析】** 与全 1 的二进制数做"与"运算,结果不变;与全 0 的二进制数做"与"运算,结果为 0。所以,设一个数,在二进制形式中,将要保留的部分置 1,不保留的部分置 0,和它做"与"运算,就是所求结果。对本题,设这个数为 255,低 8 位全 1,与它做"与"运算保留低 8 位,再将原数右移 8 位,第 2 个 8 位成为新的低 8 位,再与 255 做"与"运算……整型变量用 unsigned int 说明。

**【算法描述】** 设待分离的整数为 data。

① templt=255;

② data1=data& templt,data=data>>8;

③ data2=data& templt,data=data>>8;

④ data3=data& templt,data=data>>8;

⑤ data4=data& templt,data=data>>8;

⑥ 从高位到低位输出 data4,data3,data2,data1;

⑦ 结束。

**【测试指南】** 测试,设计输入,要知道输出才能验证结果是否正确,所以测试用例要知道输入数据的每一个 8 位的十进制是多少才行。输入一个小于 255 的数,它只占低 8 位。设最低 8 位和次低 8 位的十进制数分别为 a,b,则构造该数为(b<<8)+a,左移 8 位,变成高一级的 8 位。

**【问题扩展】** 请编写程序,输入 4 个小于等于 255 的正整数,将它们看做一个 4 字节无符号整数从左到右的 4 个 8 位,构造这个无符号整数。

## 2.3 实验 3 运算的流程控制

**实验目的**

(1) 理解程序的执行顺序。

(2) 理解程序的控制结构。

(3) 掌握分支、循环结构的使用方法。

(4) 掌握一维数组的使用。

(5) 掌握累加、累积运算的程序设计方法。

## 2.3.1 计算 π 的近似值

将 arctg(x)在 x=0 处展开,得

$$\operatorname{arctg}(x) = x - \frac{x^3}{3} + \frac{x^5}{5} - \frac{x^7}{7} + \cdots = \lim_{i=1}(-1)^{i+1}\frac{x^{(2i-1)}}{2i-1}$$

当 x=1 时,arctg(x)=π/4,从而这个级数既可以计算 arctg(x)的近似值,又可以计算 π 的近似值:

$$\pi = 4\left(1 - \frac{1}{3} + \frac{1}{5} - \frac{1}{7} + \cdots\right)$$

利用上式编程计算 π 的近似值,精确到小数点后 8 位(通项的绝对值小于 1.0E-9)。

**【本题目的】** 循环、while 循环的使用、通项的构造、求和运算的基本方法、精度控制。

**【问题分析】** 本题的关键是构造级数的通项。分析通项,分母为奇数,(2n-1),n=1,2,3,…时,就是通项的分母。求倒数就是通项的绝对值。通项中还有一个问题是符号,是正负交替的,数学上常写成$(-1)^{n+1}$,n=1,2,3…。而负数乘法还有一个特点是“负负得正”。(-1)乘一次(-1)变为正的,再乘一次(-1)又变为负的……可以实现正负交替。

关于小数点后面 8 位的精度,其实在此 π 的精确值是未知的。由于通项是逐项累加的,级数的和越来越接近精确值,所以当相邻的两个和的差小于 1.0E-9 时,就认为达到了精度要求。本题循环次数不能确定,精度是确定是否继续循环的依据,所以用 while 循环实现。

要显示小数点后面 8 位,在输出前添加下列两条语句:

```
cout.setf (ios::fixed);
cout.precision (8);
```

**【算法描述】** 设 pai 是累加和,初始为 0,u 是通项,sign 表示符号,初始为-1,eps=1.0e-9。

① pai=0,u=0,sign=1
② n=1,
③ u=1/(2n-1)　　　　//通项的绝对值
④ pai=pai+sign * u;　　　　//构造通项并累加
⑤ sign=sign * (-1)　　　　//构造下一项的符号
⑥ n=n+1　　　　//n 加 1 以便计算下一通项
⑦ 如果 u>eps,转③;否则执行下一步;　　//不满足精度,进行下一循环
⑧ 输出 4.0 * pai。

**【编程指导】** 本题编程需注意混合类型的运算,特别是整型、实型混合运算及整型数的除法。另外一点别忘记级数的计算后,乘以 4。

**【问题扩展】** 本题没有输入,可以将精度作为输入,看不同精度要求时的运行时间。还可以在计算一次 pai 后,将 n,u,pai 显示出来,查看随着精度的提高,收敛速度如何。

逐步近似是计算机求解问题的常用策略。给一个初始值,在此基础上计算出一个更

好的近似解。如果该解不满足要求,再计算一个更好的解……在级数求和中,关键是构造通项。通项不一定写成一个式子,可以逐步构造,这样会更清晰、更简单,不易出错。

## 2.3.2　比较字符串大小

比较两个字符串的大小。用户输入两个不含空格的字符串,比较它们的大小,并显示出来。例如输入: debug　debate,显示: debug>debate。

**【本题目的】** 理解字符数组、字符串,掌握通过字符数组进行字符串操作的基本方法。

**【问题分析】** 字符串的大小是比较它们在字典中的顺序。在前的小,在后的大。这实际是从头按顺序比较各对应字符在字母表中的顺序,也即 ASCII 字符的顺序。若相等则看下一个,直到对应位置的字符不等或有一个结束。由于不知道两个字符串多长,比较到哪个字符为止,所以通常用 while 循环。字符在计算机中是一个 8 位的整数,可以直接比较。字符串是否结束,就看是否结束符'\0'。

**【算法描述】** 设两个字符串为 str1[100]和 str2[100],长度不超过 99。

① 输入字符串 str1 和 str2;

② i=0

③ 如果 str1[i]==str2[i] 并且两个字符串都没有结束,执行④;否则转⑥

④　i++

⑤ 转③

⑥ flag=str1[i]-str2[i]

⑦ 如果 flag>0 显示 str1>str2

否则,如果 flag<0,显示 str1<str2

否则显示 str1=str2

⑧ 结束。

**【测试指南】** 本例需要仔细测试,因为可能的输入很多。如两个完全不同的单词,两个完全相同的单词,两个相同的单词但大小写不同,两个长短不同的单词,两个前缀相同的单词等。测试时不能认为: 若一组数据结果正确,则程序就是正确的。

**【问题扩展】**

① 你写的程序区分大小写吗? 如果区分,再写一个不区分大小写的。或反过来。区分大小写就是说"a"和"A"不同。

② 字符串的操作,一般都是逐个字符进行处理的,何时结束,就看是否有结束符'\0'。

## 2.3.3　找回文数

整数里也有回文如 12321,正反顺序的数字都是 1,2,3,2,1,称为回文数。(1)用户输入一个整数,判断是否回文数。有些回文数同时又是一个数的平方,如 676 是回文数,又是 26 的平方,称为平方回数。(2)编程找出十万以内的平方回数。

**【本题目的】** 整数的分离与构造,循环、数学函数的应用,问题的探究。

**【问题分析】** 人工判断回文数,看前后对称位置的数字是否相同,那就要找出对称位

置的数字。分离数字可以使用求余的方法，如一个数除 10 求余就是个位数字。把各位的数字放在一个一维数组 A 中，如果有 N 位，那就看 A[i]和 A[N－1－i](i＝0,…,N/2)是否相等。

回文数的特点是倒过来的数与原数是相等的，例如 1642 倒过来是 2461，不等，因此不是回文数；而 121 就是回文数。所以另一种判别回文数的方法是分离各位数字，构造一个倒过来的数，如果和原数相等，就是回文数。

至于平方的问题，可以使用系统的库函数 sqrt。

找十万以内的回文数，只要从 1 开始到十万逐个检查是否回文数并且是某数的平方即可。

**【算法描述】** 判断回文数。

① 输入整数 number；

② N1＝number；N2＝0；

③ 当 N1！＝0 时循环执行下列计算，否则转④；

```
d=N1%10;              //取个位数字
N2=N2*10+d;           //逐步构造新数
N1=N1/10;             //去掉原来的个位
```

④ 如果 N2＝number，则是回文数，否则不是；

⑤ 显示结果；

⑥ 结束。

**【编程指导】** 查找十万以内的平方回文数，循环次数已知，可以用 for 循环，这时 number 不再需要输入。

**【问题扩展】**

① 算法中第三步构造整数的方法是常用的，例如：

$$1234=((1*10+2)*10+3)*10+4$$

同学们要学会这种方法。教材习题 3 第 13 题，通项也可用这种方法构造。

② 算法第二步的 N1＝number 的作用是什么，去掉它有什么影响？

③ 请使用【问题分析】中的第 1 种方法判断是否回文数。

④ 探索立方回文数、4、5、6 等次方回文数。

⑤ 教材习题 3 第 17 题，如何判断回文诗呢？

## 2.3.4 整数的素数分解

任意一个大于 1 的正整数可以表达为一系列素数的乘积，这样的分解是唯一的，称为素数分解。例如，60 可以分解为 2×2×3×5。编写程序，显示用户输入的一个正整数的素数分解。

**【本题目的】** 练习循环，分支，数的分解方法，问题探究。

**【问题分析】** 本题有两个关键，一是数的分解，二是找下一个素数。

进行数的素数分解时，首先应找最小的素数，看能否整除。如果能，除去后还要检查能否再一次整除；如果不能，检查下一个素数，所以实际需要一个从小到大的素数列表。

构造这个列表的方法之一是“筛选法”：定义一个较大的一维数组A，数组元素的值就是它的下标，即A[i]=i，然后从A[2]开始，将其后2的倍数的元素置0；然后向后检查，找到下一个不是0的元素，再将其后该元素倍数的元素置0；继续找下一个不为0的元素……直到检查到A[N/2](N是数组大小)，则数组中不为0的元素即为素数。

**【算法描述1】** 构造素数列表。

① 定义一维数组A[N]，A[i]=i，i=0，…，N-1
② i=2
③ 如果i<N/2，执行下一步，否则转⑩
④ 　　如果A[i]=0，转⑨
⑤ 　　j=i*2
⑥ 　　如果j<N，执行下一步，否则转⑨
⑦ 　　　　a[j]=0，
⑧ 　　　　j=j+i，转⑥
⑨ 　　i=i+1，转③
⑩ 结束。从A[2]开始，非0元素是素数。

**【算法描述2】** 分解。

① 输入整数m
② i=2
③ 如果i<=m，执行下一步，否则转⑩
④ 　　如果A[i]!=0，执行下一步，否则转⑨
⑤ 　　　　若m%A[i]==0，执行下一步，否则转⑨
⑥ 　　　　　　输出A[i]
⑦ 　　　　　　m=m/A[i]
⑧ 　　　　　　转⑤
⑨ 　　i=i+1，转③
⑩ 结束。输出的序列是m的因子，m为素数时，输出本身。

**【编程提示】** ①数组的大小可定义为99999，对已知次数的循环，用for。程序也分两大块编写，一是构造素数表，二是分解。

**【测试指南】** 应找一些分解形式不同的数进行测试，如8是三个因子均为2，11是素数，98765是较大的5位数，987654是6位数等。

**【问题扩展】**

① 判别一个数是否素数与构造素数表有所不同。判别是否素数，依据是有无除1和它本身外的因子。那就要从2开始，一个一个除一下看能否整除。若这个数是n，简单的是除到n-1为止，但有必要吗？除到多少就可以了呢？

② 为了在一次运行中用户可以多次输入整数并分解，将程序写在下列循环体中：

```
while(1)
{
    <循环体>
}
```

这样的循环起什么作用呢?

## 2.4 实验4 复杂信息的表达与处理

**实验目的**

(1) 熟练掌握一维、二维数组的使用。

(2) 熟练掌握字符数组、字符串的操作。

(3) 掌握结构体的使用。

### 2.4.1 矩阵转置

编写程序,用户输入一个N(N≤10)阶方阵,将方阵转置后输出。例如:

$$\begin{bmatrix}5 & 6 & 7 & 9\\2 & 8 & 5 & 4\\3 & 7 & 16 & 15\\1 & 4 & 8 & 11\end{bmatrix} \qquad \begin{bmatrix}5 & 2 & 3 & 1\\6 & 8 & 7 & 4\\7 & 5 & 16 & 8\\9 & 4 & 15 & 11\end{bmatrix}$$

转置前方阵A　　　　转置后方阵A

要求,只使用一个数组。

**【本例目的】** 数组下标的操作。

**【问题分析】** ①方阵的阶数是不定的,但不超过10,所以应定义一个10*10的二维数组。②为确定运行时的维数,可以要求用户输入矩阵的阶数。③矩阵的转置,就是行列互换。具体到元素的变化,就是a[i,j]和a[j,i]互换。但要注意,下三角和上三角互换,编程时应注意下标的范围。变过来又变回去是常犯的错误。

**【算法描述】** 转置。设数组用A表示,$a_{ij}$表示i行j列的元素。

① 输入阶数N。

② 按行输入矩阵的元素。

③ 对i=0,…,N−1　　　　//表示行

　　对j=i+1,…,N−1　　　　//表示列

　　　　$tmp = a_{ij}$

　　　　$a_{ij} = a_{ji}$

　　　　$a_{ji} = tmp$

④ 按行显示矩阵元素。

⑤ 结束。

**【编程指导】** 输出时,矩阵元素要按列对齐,一行中的数据间用'\t'分隔,行间用'\n'分隔。

**【问题扩展】**

① 矩阵的编程要清楚地知道哪个下标表示哪个元素,下标的变化范围与矩阵元素之间的对应关系。

② 设有一个有序的整型数组，数据元素从小到大排列，初始时数组中没有元素。用户从键盘输入若干个整数，将其插入到数组的合适位置，使数组保持有序，并打印插入后的元素。程序要考虑对数组满的情况的处理。

## 2.4.2　用一维数组实现矩阵相乘

用户输入 $A_{M\times N}$，$B_{N\times K}$ 两个矩阵的元素，计算它们的乘积并输出。其中 M，N，K 也由用户输入，它们均不超过 20。要求用一维数组实现。

**【本例目的】** 用一维数组表示矩阵元素，了解这样做的好处。

**【问题分析】** 本题的关键是理解矩阵用一维数组表示时，原来 i 行 j 列的元素，在一维数组中的下标。若矩阵 A 是 M 行，N 列的，一维数组 C 的大小大于 M * N。将矩阵 A 的元素按行存放在一维数组 C 中，经过分析，可以得到它们之间的对应关系：

$$A[i][j]=C[i*N+j],\quad i=0,\cdots,M-1,\ j=0,\cdots,N-1$$

这样，只要把原来的二维数组，换为一维数组并修改下标即可。

**【问题扩展】**

① 本题若使用二维数组实现，比较清楚、直观。但如果定义的二维数组是 20×20 的，虽然能容纳 400 个元素，但要存放 2 行 21 列的矩阵就无法表示。而如果是大小为 400 甚至 200 的一维数组，能容纳的元素的个数没有增加，却容易表示 2 行 21 列，甚至 2 行 100 列的矩阵。

② 矩阵用一维数组存储，判断矩阵是否对称矩阵。

③ 对称矩阵的上三角和下三角部分是对称相同的（即 $a_{ij}=a_{ji}$）。如果保存两个相同的元素，显得有点浪费。试想办法用一维数组只存储上三角或下三角部分，但又要能顺利存储每一个元素。

## 2.4.3　反转字符串

编写程序，用户输入一个英文字符串，将其中的字符顺序反转过来（仍保存在原来的字符数组中），然后输出。例如，输入“student”，输出“tneduts”。

**【本例目的】** 字符串的操作，牢记字符串以'\0'为结束符。

**【问题分析】** 字符串的反转，从元素的变化看，就是开头的换到了末尾，末尾的换到了开头，所以，要清楚谁和谁互换，何时结束。由于字符串的长度是未知的，哪是末尾呢？所以要先写一段程序，计算字符串的长度。若长度为 N，则下标是 0 的就是开头，下标是 N－1 的就是末尾。

**【算法描述】**

① 输入字符串 str。

② 计算字符串的长度，设为 N。

③ 对 i=0，…，[N/2]－1　　　　//[]表示取整

④ 　　tmp=str[i]

　　　str[i]=str[N－1－i]

　　　str[N－1－i]=tmp

⑤ 输出 str。

**【注意事项】** 字符串以'\0'为结束符是字符串操作要牢记的。

**【问题扩展】**

① 本题是字符的互换,也可以是置换(把一个字符换为另一个字符),还可以是查找、插入、删除、替换一个或多个字符。

② 编写程序,对字符串进行加密和解密。

## 2.4.4 去掉字符串开头的多余空格

编写程序,去掉字符串开头的空格符。

**【本例目的】** 熟练掌握字符串的操作,牢记字符串以'\0'为结束符。

**【问题分析】** 空格的表示就是' '。注意,单引号中有一个空格。要去掉字符串开头的空格,先要判断开头是否有空格,将下标为 0 的字符与' '比较即可,也可以比较 ASCII 码 32。如果是空格,将后面的所有字符,包括最后的结束符'\0'全部向前移动一个位置,则第 1 个空格就删除了。使用相同的方法,删除其他空格。也可以计算开头空格的数量,将后面的字符一次向前移动多个位置。

另外,使用 cin＞＞输入,它是以空格或回车为分隔符的,空格不会存入变量中。要在输入中输入空格,应使用 cin.getline(str,100);这样的格式(其中 str 是字符数组名,100 是最大长度)。

**【算法描述】**

```
① 输入字符串,设为 str。
② i=0。
③ 如果 str[i]等于 32(' '),执行④;否则转⑨。    //
④      j=0。
⑤      若 str[j]!='\0',执行⑥;否则转⑧。
⑥           str[j]=str[j+1]                //后面的往前放
⑦           j=j+1,转⑤
⑧ 转③。
⑨ 输出字符串 str。
⑩ 结束。
```

**【编程指导】** 由于空格在视觉上是空白,是看不到的,所以,输出字符串时,可以在字符串的前后各加一个其他字符,比如"|",这样输入的字符串和处理后的字符串是否含有空格就一目了然了。

**【测试指南】** 测试时应输入开头有一个空格、多个空格和没有空格的字符串进行检验。

**【注意事项】** 字符串编程中,经常遇到的问题是屏幕显示"…烫烫烫…"这样的混乱字符串,其原因是在字符串处理过程中,字符串没有结束符,这样系统就不知道显示到何时结束了。所以,在字符串的移动、替换、删除等操作时,一定要注意是否在新的字符串末尾添加了结束符。

【问题扩展】

① 如果字符串开头有 10 个空格,本题的方法要移动 10 次。请使用一次移动的方法,去掉字符串开头的空格。

② 除字符串开头的空格,有时还需要去掉字符串末尾的空格和中间的所有空格。请实现之。

## 2.4.5 事件时间表

输入若干事件的名称和一天内的发生时间(时、分、秒,24 小时制),按顺序计算相邻两件事的发生的间隔。时间的输入格式如 18 28 15,表示 18 点 28 分 15 秒,输入的事件无顺序。

**【本例目的】** 结构体的应用、字符串的处理。

**【问题分析】**

① 每个事件具有名称和时间两个特征。名称自然使用字符串表示。时间本也可以用字符串表示,但计算两个事件之间的时间差不方便。所以,时间用三个整数分别表示时、分、秒。这样可以定义两个结构体,一个表示时间,一个表示事件。时间结构体的成员有三个整数,分别表示时、分、秒;事件结构体的属性由事件名称、时间和时间差组成。若干个事件用结构体数组表示,输入时可以以 0 0 0 0 表示结束。

② 排序。时间用三个整数表示,排序时要先比较“时”,“时”小的在前。如果“时”相等,“分”小的在前;“分”相等,“秒”小的在前。还要注意排序时交换元素要整个结构体交换。

③ 时间差。可以先将时间换算为秒,相减,再将结果换算为时、分、秒。

**【算法描述】** 设:

时间结构体:

```
    struct TIME
    {   int Hour, Minute,Second;   };
```

事件结构体:

```
    struct EVENT
    {
    char Name[20];
    struct TIME Time;
    struct TIME Distance;
    };
```

事件数组:EVENT event[50];

事件个数 N。

(1) 比较和排序

```
对 i=0,…,N-2
    对 j=0,...,N-2-i
        如果 (event[j].Time.Hour>event[j+1].Time.Hour)            或
```

```
(event[j].Time.Hour==event[j+1].Time.Hour   并且
 event[j].Time.Minute>event[j+1].Time.Minute)      或
(event[j].Time.Hour==event[j+1].Time.Hour   并且
 event[j].Time.Minute==event[j+1].Time.Minute 并且
 event[j].Time.Second>event[j+1].Time.Second   )
     tmp=event[j]                //tmp是EVENT型变量
     event[j]=event[j+1]
     event[j+1]=tmp
```

(2) 计算时间差

```
itmp0=event[0].Time.Hour*3600+event[0].Time.Minute*60+event[0].Time.Second
对 i=1,...,N-1
        itmp1=event[i].Time.Hour*3600+event[i].Time.Minute*60+event[i].
                Time.Second
        itmp=itmp1-itmp0
        event[i].Distance.Hour=itmp/3600
          event[i].Distance.Minute=(itmp%3600)/60
          event[i].Distance.Second=???              //请同学们自己写此式
        itmp0=itmp1
```

其中 itmp,itmp0,itmp1 均为整数。

**【编程指导】** 本题略显烦琐,请同学们一步一步做,不要急于求成。整个程序的步骤是:

① 设计数据结构:结构体、变量,各表示什么信息。

② 输入。

③ 排序。

④ 计算时间差。

⑤ 显示结果。

注意,这5步一定一步一步做,不要在一步中急于实现其他功能。

**【测试指南】** 测试应就一个事件、两个事件、多个事件,以及相同和不同的时、分、秒时间进行。

**【问题扩展】**

① 时间的处理,体现了不同的数制或信息的表示之间的转换关系,具有一定的代表性。实现方法也不唯一。同学们可以尝试其他方法,比如不同的排序方法、比较方法和计算时间差的方法。

② 常见的时间格式之一是:时:分:秒,例如16:08:12。如果用户输入的是这种格式,本题又如何实现呢?

③ 如果时间格式不是24小时制,而是12小时制呢?

④ 如果发生的时间不是一天之内的,而是一月之内,一年之内的呢?

⑤ 本题的实现不是一个简洁的方法,还有更好的方法吗?

# 2.5 实验 5 划分模块 逐层求解——函数

**实验目的**

（1）掌握基本函数的定义、调用和声明。

（2）掌握函数参数的值传递和引用传递。

（3）掌握函数与数组、函数与字符串的组合使用方法。

（4）掌握函数重载和递归函数的使用。

## 2.5.1 编写求一元二次方程的根的函数

编写函数求一元二次方程 $ax^2+bx+c=0$ 的根并显示，要求判断是方程式实根、重根还是复根，由主函数传递三个系数值给该函数来计算方程的根。

**【本例目的】** 练习函数的基本定义、调用和声明的格式。

**【问题分析】** 当 $a\neq0$ 时，一元二次方程的求根公式为：

$$x=\frac{-b\pm\sqrt{\Delta}}{2a},\quad 其中\ \Delta=b^2-4ac$$

要分条件分解这个方程的求根过程：如果 $a=0,b=0,c=0$，那么是任意根；如果 $a=0,b=0,c\neq0$，那么无根；如果 $a=0,b\neq0$，那么有单根 $-c/b$。只有当 $a\neq0$ 时才可用上述公式求出两个根。而且，若 $\Delta>0$，则是两个实根；若 $\Delta=0$，则是一个重根；若 $\Delta<0$，则是两个复根。

**【算法描述】** 见主教材第 3 章。

**【编程指导】** 函数声明的格式如下：

```
void Root(double a, double b, double c);
```

主函数的格式如下：

```
int main()
{
    请输入三个实系数
    调用 Root 函数

    return 0;
}
```

**【测试指南】** 请输入不同情况的三个实数来验证程序。

**【问题扩展】** 使用二分法、弦截法和牛顿迭代法求方程的根。

## 2.5.2 编写函数求一元 n 次多项式的值

给定 n+1 个实系数：$a_n$、$a_{n-1}$、…、$a_1$、$a_0$ 以及 x 的值，计算它们构成的多项式 $a_nx^n+a_{n-1}x^{n-1}+\cdots+a_1x+a_0$ 的值，要求编成一个函数，系数和自变量 x 为参数。

【本例目的】 练习数组作为参数在函数中的使用。

【问题分析】 采用秦九韶算法。秦九韶算法是中国南宋时期的数学家秦九韶提出的一种多项式简化算法，在西方被称作霍纳算法。一元 n 次多项式的求值一般需要经过[n(n+1)]/2 次乘法和 n 次加法，而秦九韶算法只需要 n 次乘法和 n 次加法，可以以更快的速度得到结果，减少了 CPU 运算的时间。

把这个一元 n 次多项式作如下形式的改写：

$$
\begin{aligned}
& a_n x^n + a_{n-1} x^{n-1} + \cdots + a_1 x + a_0 \\
= & (a_n x^{n-1} + a_{n-1} x^{n-2} + \cdots + a_1) x + a_0 \\
= & ((a_n x^{n-2} + a_{n-1} x^{n-3} + \cdots + a_2) x + a_1) x + a_0 \\
= & \cdots\cdots \\
= & (\cdots\cdots((a_n x + a_{n-1}) x + a_{n-2}) x + \cdots + a_1) x + a_0
\end{aligned}
$$

在进行多项式求值时，先计算最内层括号内一次多项式的值，即

$$v_1 = a_n x + a_{n-1}$$

然后由内向外逐层计算一次多项式的值，即

$$
\begin{aligned}
v_2 &= v_1 x + a_{n-2} \\
v_3 &= v_2 x + a_{n-3} \\
&\cdots\cdots \\
v_n &= v_{n-1} x + a_0
\end{aligned}
$$

这样，求一元 n 次多项式的值的问题就转化为求 n 个一次多项式值的问题。

【算法描述】

开始

① 输入多项式的次数。

② 输入多项式的 n+1 个系数和 x 的值。系数存放在数组 v 中，v[i]为 $a_i$。

③ i=n，v=a[i]。

④ 计算 v=vx+a[i-1]。

⑤ i--，如果 i>0，转④。

⑥ 输出结果。

结束

【编程指导】 函数的声明的格式如下：

```
double Poly(double a[], int n, double x);
```

主函数的格式如下：

```
int main()
{
    输入多项式的系数 n
    输入数组的 n+ 1个元素和 x的值
    调用 Poly函数
    输出结果
```

```
    return 0;
}
```

【问题扩展】　编写函数计算二元多项式的值。

### 2.5.3　编写函数去掉任意一个字符串头部和尾部的空格

编写函数去掉任意一个字符串头部和尾部的空格。编写主函数,输入字符串,显示处理前的字符串,调用函数,再显示处理后的字符串。由于空格的显示是不可见的,显示字符串时可分别在字符串的前面和后面各加一个“|”。

【本例目的】　练习字符串作为参数在函数中的使用。

【问题分析】　去掉末尾的空格,只要将最后一个不是空格的符号的后一个符号设置为'\0';去掉字符串前面的空格要将不是空格的字符逐个前移,包括结束符'\0'。

首先分别编写去掉一个字符串左侧空格和右侧空格的函数,然后由主函数依次调用它们完成去掉字符串左右的全部空格。

判断空格时可以使用空格字符' ',也可以使用它的 ASCII 编码值 32。应注意字符串的结尾符为'\0'或 0。

【算法描述】

开始

① 定义字符串变量,并输入其值

② 下标位置=0

③ 当下标位置的字符是空格时循环

　　下标位置加 1

④ 将当前下标位置开始的字符串复制到原字符串的开始位置

　得到去掉左侧空格的字符串

⑤ 下标位置=0

⑥ 当下标位置的符号不是结束符时循环

　　下标加 1

⑦ 下标减 1

⑧ 当下标位置的字符是空格时循环

　　下标减 1

⑨ 下标加 1

⑩ 将当前下标位置的字符置为结束符,得到去掉右侧空格的字符串

结束

【编程指导】　去掉字符串左侧空格函数的声明格式如下:

```
void ltrim(char a[]);
```

去掉字符串右侧空格函数的声明格式如下:

```
void rtrim(char a[]);
```

主函数的定义格式如下：

```
int main()
{
    请输入字符串
    调用 ltrim 函数去掉字符串左侧的空格
    调用 rtrim 函数去掉字符串右侧的空格
    显示去掉左右空格后的字符串

    return 0;
}
```

**【测试指南】** 通过输入5类字符串来进行测试，即仅有左侧空格的字符串、仅有右侧空格的字符串、左右侧都有空格的字符串、中间有多处空格的字符串、没有空格的字符串。

**【问题扩展】** ①如何去掉字符串中间的空格；②如何去掉字符串的全部空格；③如何去掉字符串中的任一个字符。

## 2.5.4 数组的转换

分别编写整型到实型、实型到整型、整型到字符、字符到整型的数组的转换函数，数组的大小相同，使用重载实现，其中字符与整数的转换仅是类型的转换，它们的值相同。编写主函数，验证函数的转换功能。

**【本例目的】** 重载函数的编写。

**【问题分析】** 定义四个同名函数。第一个函数转换整数数组到浮点数数组，其形参是一个整型数组、一个浮点型数组以及它们的元素个数；第二个函数转换浮点数组到整型数组，其形参是一个浮点数数组、一个整数数组以及它们的元素个数；第三个函数转换整数数组到字符数组，其形参是一个整数数组、一个字符数组以及它们的元素个数；第四个函数转换字符数组到整数数组，其形参是一个字符数组、一个整数数组以及它们的元素个数。

**【算法描述】**

开始

循环将第一个数组的每一个元素赋值给第二个数组的对应位置的元素

结束

**【编程指导】** 转换整数数组到浮点数数组的函数声明格式如下：

```
double ArrayCopy( int a[], double b[], int n);
```

转换浮点数数组到整数数组的函数声明格式如下：

```
double ArrayCopy( double a[], int b[], int n);
```

转换整数数组到字符数组的函数声明格式如下：

```
double ArrayCopy( int a[], char b[], int n);
```

转换字符数组到整数数组的函数声明格式如下：

```
double ArrayCopy(char a[], int b[], int n);
```

主函数的格式如下：

```
int main()
{
    分别定义整型数组、双精度数组和字符数组
    输入其实际大小和各个元素的值
    分别调用以上四个函数并显示结果

    return 0;
}
```

**【测试指南】** 首先输入数组的大小，然后输入任意两种类型的数组元素值。

**【问题扩展】** 如何完成不同类型的二维数组的转换？

## 2.5.5 递归实现级数求和

编写函数，计算级数 1/1+1/2+1/3+…+1/n 的和，n 为参数，要求用递归实现。编写主函数，输入 n，调用函数计算级数的和，在主函数中显示结果，要求保留 8 位小数。

**【本例目的】** 练习递归函数的编写。

**【问题分析】** 把该问题作为递归问题来考虑，其基本形式为：n 为 1 时的结果为 1/1；

一般地，先将公式 1/1+1/2+1/3+…+1/n 分为两个部分：1/1+1/2+1/3+…+1/(n−1)和 1/n；再将公式 1/1+1/2+1/3+…+1/(n−1)分为两个部分：1/1+1/2+1/3+…+1/(n−2)和 1/(n−1)；依次类推，不断进行递归即可。

**【编程指导】** 递归函数的定义格式如下：

```
double sum(int n)
{
    如果 n为 1返回 1
    否则分解为 1/n和 n-1项之和,进行递归
}
```

主函数定义格式如下：

```
int main()
{
    输入 n
    调用函数 sum求级数和
    输出结果

    return 0;
}
```

【问题扩展】

① 编写计算阶乘的递归函数,阶乘的公式为:n!=1×2×…×n;

② 编写计算斐波那契数列的递归函数,斐波那契数列公式为:

$$f(0)=1,\quad f(1)=1,$$
$$f(n)=f(n-1)+f(n-2)。$$

## 2.5.6 求数组元素的最大值的递归函数

编写函数求数组元素的最大值,要求用递归实现。编写主函数检验函数的功能。

【本例目的】 练习递归函数的编写。

【问题分析】 假设数组名为 a,把该问题作为递归问题来考虑,其基本形式为:n 为 1 时的结果为 a[0];

一般地,先将 a[0]、a[1]、…、a[n-1]分为两个部分:a[0]、a[1]、…、a[n-2]和 a[n-1];再将公式 a[0]、a[1]、…、a[n-2]分为两个部分:a[0]、a[1]、…、a[n-3]和 a[n-2];依次类推,不断进行递归即可。

【编程指导】 递归函数的定义格式如下:

```
int max( int a[], int n)
{
    如果 n 为 1 返回 a[0]
    否则分解为 a[0]、a[1]、…、a[n-2]和 a[n-1],求它们的最大值,进行递归
}
```

主函数定义格式如下:

```
int main()
{
    输入 n 和数组 a 的 n 个元素的值
    调用函数 max 求最大值
    输出结果

    return 0;
}
```

【问题扩展】

① 编写在一个整数数组中查找一个值是否存在的递归函数。

② 编写计算 arcsin(x)的递归函数,精度要求达到 $10^{-7}$。arcsin(x)公式如下:

$$\arcsin(x)=x+\frac{x^3}{2\cdot 3}+\frac{1\cdot 3\cdot x^5}{2\cdot 4\cdot 5}+\cdots+\frac{(2n)!\,x^{2n+1}}{2^{2n}(n!)^2(2n+1)}+\cdots$$

## 2.5.7 随机生成整副 54 张扑克牌的函数

编写函数,生成一副扑克牌,有 54 张,顺序和花色是随机的。编写主函数,调用函数生成扑克牌,在主函数中显示 54 张牌,要显示出花色和牌点。

**【本例目的】** 系统函数的使用。

**【问题分析】** 54 张扑克牌分别用以下整数表示：黑桃 A～K 用 0～12 表示，红桃 A～K 用 13～25 表示，方块 A～K 用 26～38 表示，梅花 A～K 用 39～51 表示，大小王用 52、53 表示。

C++ 系统提供了三个与随机数有关的函数。

(1) 获取当前日历时间函数，其声明格式为 time_t time(time_t *);。

其中 time_t 是定义在 ctime 中的一个类型，表示一个日历时间，也就是从 1970 年 1 月 1 日 0 时 0 分 0 秒到此时的秒数，原型是：typedef long time_t;。

time_t 其实是一个长整型，由于长整型能表示的数值有限，因此它能表示的最迟时间是 2038 年 1 月 18 日 19 时 14 分 07 秒。

(2) 初始化随机数发生器函数，声明格式：void srand(unsigned seed);。

(3) 产生一个 0～32767 范围内的随机数的函数，声明格式：int rand();。

以上两个函数定义在头文件 cstdio 中。

以上三个函数应结合起来使用，比如：

```
srand(time(null));
int x= rand();
```

对于本例，应产生 0～53 范围内的随机数，x＝0＋rand()％54。

**【算法描述】**

开始

① 定义大小为 54 的整型数组，初值为每个下标的值。

② 生成下一个随机数。

③ 判断是否与以前生成的随机数重复：

　　是，则重新生成随机数，再次判断是否重复；

　　否，则将其加入数组。

④ 转②，直到生成 54 张牌。

结束

**【编程指导】** 本题生成 54 个不相重复的随机数是重点，也是难点。每次生成一个新的随机数就需要判断其是否与之前产生的随机数重复。

扑克牌生成函数的声明格式如下：

```
void Puke( int a[], int n);
```

主函数定义格式如下：

```
int main()
{
    定义整型数组
    设置数组的初始值
    调用函数 Puke()生成一副扑克牌
    输出结果
```

```
    return 0;
}
```

【问题扩展】 对于6个人参与的扑克牌游戏,如何随机生成每个人的9张牌?

## 2.5.8 验证哥德巴赫猜想

1742年法国数学爱好者哥德巴赫在给著名数学家欧拉的信中提出了“哥德巴赫猜想”:任何一个大于等于4的偶数均可以表示为两个素数之和。编写程序,对4到用户输入的整数M之间的所有偶数验证哥德巴赫猜想是否成立。

【本例目的】 练习函数的综合应用与结构化、模块化程序设计的方法。

【问题分析】 “哥德巴赫猜想”是数论中的一个著名难题,200多年来无数数学家为其呕心沥血,却始终无人能够证明或伪证这个猜想。现在可以使用计算机来验证。

第1步 提出拟解决的问题,即“验证哥德巴赫猜想”。

第2步 在主模块(main)中输入一上限数M,验证从4到M的所有偶数是否能被分解为两个素数之和。

第3步 将“如何验证X是否能被分解为两个素数之和”设计为一个模块(isPrimeSum)。

第4步 将“一个数是否为素数”设计为一个模块(isPrime)。

【算法描述】

(1) 判断一个数是否为素数

对一个整数a

用[2,a-1]中的每一个整数去除该数,如果余数均不为0,则说明该数为素数。

(2) 判断一个数是否可分解为两个素数之和

设该数为x,

对[2,a-1]中的每一个数p以及x-p,判断它们是否均为素数。

如果有一组这样的数对,就是可以分解。

如果所有的数对都不是素数,说明哥德巴赫猜想不成立。

【编程指导】 判断一个数是否为素数的函数isPrime()声明格式如下:

```
bool isPrime( int p);
```

判断一个数是否可分解为两个素数之和的函数isPrimeSum()的定义格式为:

```
bool isPrimeSum( int x ){
    调用函数 isPrime 依次判断 p 和 x-p 是否都是素数
}
```

本例需要输入一个较大的数M,然后从4到M对每一个偶数循环调用isPrimeSum函数。主函数定义格式如下:

```
int main()
{
```

```
    定义正整数 M
    输入 M
    循环调用函数 isPrimeSum 判断小于 M 的偶数 x 是否可分解为两个素数之和
    输出结果

    return 0;
}
```

**【问题扩展】** 设计字符界面的计算器程序，包括两个数的加、减、乘、除运算的函数，主函数调用这些函数。

# 2.6 实验 6 指针的应用

**实验目的**

(1) 理解地址和指针的含义、指针的类型以及指针变量的概念。

(2) 掌握指针变量的声明及其初始化、赋值等方法。

(3) 掌握有关指针变量运算的方法。

(4) 熟悉指针在字符串、函数调用及复杂数据类型中的典型应用。

(5) 熟悉动态数组的使用及函数指针的使用。

本节实验均采用指针编程方式实现。

## 2.6.1 将字符串形式的时间转换为毫秒

如果稍加留意，就会发现在 MP3 音乐文件旁边经常会有一个同名的、后缀为 lrc 的文件，那就是歌词文件。为了使得歌词能随着音乐播放同步显示出来，在 lrc 文件中有很多时间标签用来说明何时显示文字。比如“01:02.21”就是一个典型的标签形式，它表示在“1 分 2.21 秒”的时刻显示该标签后的文本。

本题目的任务是：编写程序读取用户输入的典型 MP3 时间标签，转化为毫秒的形式输出。典型 MP3 时间标签形式为 mm:SS.rr，其中 m、s 和 r 都是一位数字，mm 表示分，SS.rr 表示秒，rr 为秒的十进制余数。

**【本题目的】** 熟悉利用指针操作字符串的方法，熟悉字符数字和普通数字的转化方式。

**【问题分析】** 该问题的核心是要去掉字符串中的符号“:”和“.”，将其他部分字符串分别转化为代表分和秒的整数。将字符转化为整数的方法很多，最基本的方法是逐位转化。若 ch 为一位数字字符，则其对应普通整数为 ch－'0'。

**【算法描述】**

① 取第 0 个字符，转换为数字后乘以 10，存入变量 m； //m 存储分

② 取第 1 个字符，转换为数字后，叠加到变量 m 上，再将 m 转化为毫秒；

③ 取第 3 个字符，转换为数字后乘以 10，存入变量 s； //s 存储秒

④ 取第 4 个字符，转换为数字后，叠加到变量 s 上，再将 s 转化为毫秒；

⑤ 取第 6 个字符,转换为数字后乘以 100,存入变量 r;  //r 存储秒的小数部分

⑥ 取第 7 个字符,转换为数字后乘以 10,叠加到变量 r 上;

⑦ 将 m、s 和 r 相加,就得到时间的毫秒值;

⑧ 输出结果。

【结果示例】

```
输入:00:28.60
输出:28600
```

【测试指南】 本题目测试数据在 60 分钟以下,超过这个范围,程序可能出现异常情况。另外,时间标签中间没有空格,若出现空格(比如在冒号附近),也会引起错误。

【问题扩展】 如果时间标签中间有空格,是否有办法修正?请尝试一下。

## 2.6.2 将整数变换为以","号分隔的形式

在现实中,常常见到一个大整数以逗号分隔的形式显示。例如,整数 12345678 可以以 12,345,678 的形式显示,显示的方法是将数字从右向左每隔三位加一个逗号。

本题目要求实现的功能是:读取用户输入的一个整数,转换为以逗号分隔的字符串形式,而后在屏幕上输出此字符串。

【本题目的】 熟悉整数的按位分离的相关技巧,练习将整数转换为字符的方法,学习运用指针操作字符串的方法。

【问题分析】 为了把整数转换为以","号分隔的字符串,可以将整数逐位分离并转化为数字字符,而后写入一个字符串中,并在每输出三个字符后,加入一个逗号。需要注意的是,当输出三个数字字符后,整数恰好处理完毕(即输出的三个数字字符恰好是最高位的三个数),这时候不要加逗号。

【算法描述】 算法的主要思路可描述如下:

```
//假定 str[]用于存储结果,N 为输入的整数
i=0;                                   //设定字符串 str 初始位置
当(N>0)
{
    取 N 的个位赋值给整型变量 k;
    将 k 转化为数字字符存入 str[i];
    N=N/10;
    i++;                               //位置后移
    若 N>0 且 i 是 3 的倍数,则置 str[i]为',' 且 i++;
}
在 str 尾部加上结束标志'\0';
将字符串 str 反转;                     //这样就得到正确结果了
```

**注意**:这里当 i 为 3 且 N 为 0 时,对应的情形就是"输出的三个数字字符恰好是最高位的三个数",这时候不要加逗号。

【结果示例】

```
输入：31415926
输出：31,415,926
```

【测试指南】 输入 0,123,1234,123456 等不同长度的数据试试,输入负数呢?

【问题扩展】 有的人让 N 每次对 1000 求余,这样一次就可得到三位数字。真是这样吗? 考虑 N 是数字 1000000 的情形,这时 N%1000 得到的是 0 而不是 000,而要求输出的格式是 1,000,000。因此,采用让 N 每次对 1000 求余的方法并非不可行,但是要处理类似于 N 是 1000000 的情形。这种做法不见得方便,请读者尝试一下。

## 2.6.3 用一个函数求多个实数的平均值、最大及最小值

函数的返回值用于向主调程序返回结果,比如 sin、cos 函数都是返回一个 double 型数据。但是函数中的 return 语句返回的只能是一个数据,比如返回一个 int 型、double 型或指针型数据等。如果要让一个函数返回多个结果,就不能使用 return 语句了。在前面章节提到过用引用方式可以从一个函数得到多个结果,其实用指针方式也可以实现这一点。

本题目要求实现一个函数,它接受一组实数作为参数,计算这些数的平均值、最大值及最小值,然后将这些结果返回给主调函数。在主函数中输入数据并调用这一函数,输出函数的计算结果。

【本题目的】 熟悉使用指针调用函数的方法。练习运用指针从函数中取回结果的方法。

【问题分析】 当向一个函数传入整型或实型的数组时,一般需要再传入一个整数来说明数组中实际使用的元素数目。同时,为了从函数中取回平均值、最大值及最小值,需要传入三个实型变量的地址。于是,本题目的函数有五个参数,其原形可设为如下形式:

```
void Calc(double * p, int n, double * pAvg, double * pMax, double * pMin);
```

其中,指针 p 为传入的数组地址,n 为数组中实际使用的元素数目。pAvg、pMax、pMin 为取回平均值、最大值及最小值的三个变量的地址。

【算法描述】

① 在主函数中定义平均值变量 avg、最大值变量 max 及最小值变量 min,以及其他相关的变量。

② 读取若干实数到数组 data,并记录实数个数 n。

③ 调用函数计算,调用方式为:Calc( data, n, &avg, &max, &min)。

④ 输出平均值、最大值及最小值。

【结果示例】

```
输入:
1.1  2.2  3.3  4.4  5.5  6.6  7.7  8.8  9.9
输出:
平均值 5.5  最大值 9.9  最小值 1.1
```

【问题扩展】 尝试将本题目用引用方式实现，比较引用和指针两种实现方式的差异。

### 2.6.4 二分法求方程根的通用函数

对于区间[a, b]上的连续单调函数f(x)，如果f(a) * f(b)<0，则在(a, b)中必然有一个f(x)的根。可以用二分法求f(x)的根，即每次将区间[a, b]从中点c分开，然后将[a, b]区间缩短为包含根的那一半区间；依次进行下去，直到中点c使得f(c)的绝对值足够小。

本题目要求编写一个用二分法求方程f(x)=0的根的通用函数。在主函数中测试，求方程 $x-\sin(x)-1=0$ 在[0,3]之间的根，求方程 $e^x-\sin(x)-2=0$ 在[0,2]之间的根。近似解精度要求 $|f(c)|<1.0e-5$。

【本题目的】 练习函数指针的用法。

【问题分析】 由于通用的二分法函数需要传入函数的指针以及求解区间，因此需要3个参数。不妨假定函数原形如下：

```
double process(double  x0, double  x1, double(* fun)(double));
```

这里求解区间为[x0, x1]，函数指针为fun，它所代表的函数是一元函数。显然，由于迭代过程中区间的变化，上面函数的参数和返回值都应该是double类型。

另外，对于方程 $x-\sin(x)-1=0$ 和 $e^x-\sin(x)-2=0$，也应当定义两个函数，分别计算 $x-\sin(x)-1$ 和 $e^x-\sin(x)-2$ 的值。以便于代入通用求解函数。

【算法描述】 通用二分法函数的算法要点可描述如下：

```
计算[x0, x1]的中点 x;
当(f(x)的绝对值>精度)              //循环执行(即迭代)的条件
{
    若 f(x)与 f(x0) 异号,则 x1=x;
    否则,      x0=x;
    计算[x0, x1]的中点 x;
}
```

【结果示例】

输出：

```
x-sin(x)-1=0在[0,3]的根是 root=1.93457
exp(x)-sin(x)-2=0在[0,2]之间的根 1.05412
```

【问题扩展】 编写梯形法求函数的数值积分的通用函数。

### 2.6.5 将十进制写法的IP地址转换成二进制写法

输入一个点分十进制写法的IP地址，然后将其转换为32位的二进制写法。

【本题目的】 练习使用指针操作字符串的方法，字符与数字之间的转化，十进制与二进制的转换。

【问题分析】 首先，以字符串形式读入点分十进制写法的IP地址。

然后,逐个读取字符并将其转化为数字,每次遇到'.'或结束标志时,再将这些数字转化为一个整数。

将前面得到的 4 个整数,转化为二进制形式,为了保证二进制形式的数字位数为 8 位,应再把二进制数转化为字符串形式。这部分工作最好通过函数来实现。

最后,将 4 个二进制形式的字符串连接起来即可得到结果。

**【算法描述】**

① 以字符串形式读入点分十进制写法的 IP 地址。

② 读取字符,若遇到数字字符,则将其转化为数字,继续步骤②。
若遇'.'或结束标志时,将前面几个数字合成一个整数 N。

③ 将整数 N 带入函数,得到十进制数 N 转化出来的 8 位二进制字符串 r,将二进制字符串 r 连接到结果串 bin 中。

④ 将步骤②、③执行 4 次。

⑤ 输出结果串 bin。

**【编程指导】**

```
//定义十进制转化为二进制的函数
void tran(int N, char * res)          //N为要转化的数,res保存结果二进制串
{
    strcpy(res,"00000000");           //将 res 的 8个位置全部设置为'0';
    设 j=7;
    当 (N>=1)
    {
        int i=N%2;
        N=N/2;
        char r=i+'0';                 //转化为字符
        res[j]=r;                     //从低位向高位存
        j--;
    }
}
int main()
{
    定义字符数组 r1[10],bin[40]="";//r1存一个整数转换得到的二进制
                                  //bin存最终结果
    读入点分十进制写法的 IP地址到 str[]中;
    循环 4次{
        逐个读取字符并将其转化为数字,当遇到'.'或结束标志时停止;
        再将这些数字转化为一个整数 N;
        tran(N, r1);
        strcat(bin,r1);               //将 r1连接到 bin中
    }
    输出结果 bin字符串;
}
```

**【结果示例】**

输入：64.193.3.7
输出：01000000110000010000001100000111

**【问题扩展】** 上述解法没有考虑十进制 IP 地址的合法性。如果用户输入的格式不正确或数值不在[0,255]之间，程序就会出现异常。为程序添加 IP 地址合法性校验的功能。合法时才进行转换，不合法则给出提示。

## 2.6.6 统计处理多个学生的成绩

已知有 5 个学生各有 4 门课程成绩，完成以下要求：

(1) 计算每一门课程的最高分。

(2) 计算每个学生的各门课程平均分。

(3) 分别统计每个同学不及格课程的门数。

(4) 输出平均成绩在 90 以上的学生的信息。

(5) 输出每门课程都在 85 分以上的学生的全部成绩和平均成绩。

以上每个要求都分别通过不同的函数实现，学生的成绩应以结构体形式存放。

**【本题目的】** 练习利用指针操作结构体的方法，以及用结构体数组作为参数调用函数的方式。

**【问题分析】** 由于要使用结构体存储每个人的成绩，且一共有 5 名学生，因此应使用结构体数组存储所有人的成绩，这样也便于以后作为参数传递。

存储学生成绩的结构体可定义如下：

```
struct Grade {
    char ID[10], name[40];          //ID 为学号,name 为姓名
    double math,phy,chem,eng;       //4门课分别是数学、物理、化学、英语
};
```

考察第(1)条要求，为了要计算每一门课程的最高分，显然要把结构体数组按地址传递方式传给函数，还要传递一个整数表明学生人数(即结构体数组的实际使用数量)。另外还可以指定另一个参数代表要找出最高分的课程。该函数可定义如下：

```
int FindMax(Grade * s,int n,int k);
```

其中，s 为结构体数组指针，n 为学生人数。k 为所选课程，这里可假定 k=1、2、3 或 4，分别代表数学、物理、化学、英语 4 门课程。

考察第(2)条要求，要计算每个学生各门课程的平均分，可以把结构体数组及学生人数传给函数。该函数可定义如下：

```
void Average(Grade * s,int n);
```

由于 Grade 结构体中没有平均分属性，所以每个人的平均分只能在 Average 函数中直接输出。

考察第(3)条要求，要计算每个学生不及格课程的门数，当然这里也可以像 Average

函数一样把结构体数组传给函数,其实只传递一个结构体的值也是可以的。不妨在这里尝试一下,这个函数可定义如下:

```
int Fail(Grade g);
```

这里变量 g 以值传递方式获得外部的值,统计该学生不及格课程的门数,然后将结果返回。当然,为了考察所有学生的情况,就应当在主函数中循环调用此函数,将每个学生的成绩依次传入。

考察第(4)和(5)条要求,它们和计算平均成绩的要求类似,都是遍历所有学生信息,作出相关计算,并输出有关信息。这两个函数的形式如下:

```
void FindExcel(Grade * s,int n);
void FindGood(Grade * s, int n);
```

**【编程指导】**

```
定义结构体 Grade;
int FindMax(Grade * s,int n,int k)
{
    int M=0;
    if(k==1) {                    //math 课程
        循环 i=0 到 n-1
        {   寻找并保存 s->math 的最大值;  s++;   }
    }
    if(k==2) {                    //phy 课程
        ……
    }
    ……略……
    return M;
}
void Average(Grade * s,int n)
{
    循环 i=0 到 n-1
    {
        计算 s->math,s->phy,s->chem,s->eng 的平均值存入 avg;
        cout<<s->name<<" 平均分:"<<avg<<endl;
    }
}
int Fail(Grade g)
{
    统计 g.math,g.phy,g.eng,g.chem 小于 60 分的门数,存入 sum;
    return sum;
}
void FindExcel(Grade * s,int n) { …… 略 ……  }
void FindGood(Grade * s, int n) { …… 略 ……  }
```

【结果示例】

数学最高分：90　物理最高分：98　化学最高分：99　英语最高分：98

张三　平均分：67.75
王二　平均分：88
吴一　平均分：92.25
李四　平均分：90.5
刘五　平均分：89.25

张三　不及格门数：2
王二　不及格门数：0
吴一　不及格门数：0
李四　不及格门数：0
刘五　不及格门数：0

……以下略……

# 2.7　实验7　结构抽象　数据封装——类与对象

**实验目的**

(1) 掌握类、类的成员变量和成员函数的定义格式。
(2) 掌握类的构造函数和析构函数的定义格式。
(3) 掌握类的对象的定义和使用格式。
(4) 掌握对象指针、对象数组的定义与使用方法。
(5) 掌握类的嵌套使用方法。
(6) 掌握类与多程序文件工程的使用方法。

## 2.7.1　圆类的设计及使用

设计一个表示圆的类，数据成员为半径，能计算圆的周长、面积以及和另一个圆构成的同心圆环的面积等。在主函数中用该类声明对象，验证类的功能。

**【本例目的】** 练习类、成员变量、成员函数和对象的基本使用。

**【问题分析】** 类用来描述事物的属性和功能，属性用变量、数组、对象等描述，一般作为私有成员；功能用成员函数表示，一般作为公有成员。为方便对象的初始化，通常要定义构造函数。如果需要在撤销对象前做扫尾工作，可以定义析构函数。其他根据需要，可定义成员函数。

**【算法描述】**

定义圆类

私有数据成员包括：半径；

公有的成员函数包括：

构造函数,带一个参数,初始化半径;
计算圆周长的函数,不需参数;
计算圆面积的函数,不需参数;
计算圆环面积的函数,需要一个圆类对象作参数;
析构函数;

在主函数中声明圆类的两个对象,分别计算它们的周长和面积以及它们构成的圆环的面积并显示。

**【编程指导】** 圆类的基本定义格式如下:

```
class Circle
{
private:
    私有半径成员变量
public:
    构造函数:Circle(double radius)
    计算圆周长的函数:double Area()
    计算圆面积的函数:double Length()
    计算圆环面积的函数:double RingArea(Circle c2)
    析构函数:~Circle()
};
```

主函数的格式如下:

```
int main()
{
    定义圆类的两个对象
    计算圆的周长并显示
    计算圆的面积并显示
    计算圆环的面积并显示

    return 0;
}
```

**【问题扩展】** 定义一个椭圆类,它包含长轴和短轴半径,计算周长和面积的函数、两个椭圆所包围的圆环的面积函数,以及构造函数、析构函数。

## 2.7.2 三角形类的设计与使用

用类实现三角形的功能。三角形由三个边组成,能判断三角形是否合法,是否等腰、等边、直角、钝角、锐角三角形,根据三边计算三角形的面积等。

**【问题分析】** 定义一个三角形类,包括三个边长私有成员变量、公有构造函数和析构函数,并增加判断三边是否构成三角形的函数,三边构成的三角形面积的公有成员函数,以及是否是等腰、等边、直角、钝角和锐角三角形的公有成员函数,最后在主函数中定义三角形类的一个对象并调用这些函数,显示计算结果。

三角形的类别,需根据三边即可判断。

**【本例目的】** 练习类、成员及对象的定义和使用。

**【算法描述】**

定义三角形类:

私有成员:三条边长;

公有成员函数:

构造函数,带三个参数,初始化三边长;

设置三角形的三边的成员函数;

判断三个边是否构成三角形的成员函数;

计算三个边所构成的三角形面积的成员函数;

判断是否是等腰三角形的成员函数;

判断是否是等边三角形的成员函数;

判断是否是直角三角形的成员函数;

判断是否是钝角三角形的成员函数;

判断是否是锐角三角形的成员函数;

析构函数;

在主函数中定义一个三角形的对象,初始化,判断是否构成三角形,计算三角形的面积并显示;根据用户的输入重新设置三边,计算面积,判断类别并显示。

**【编程指导】** 三角形类的基本格式如下:

```
Class Triangle
{
private:
    三个边长的成员变量
public:
    构造函数:Triangle (double a, double b, double c)
    设置参数的成员函数 void Set(double a, double b, double c)
    判断三个边是否构成三角形的函数:bool IsTriangle()
    计算三角形面积的函数:double Area()
    判断是否是等腰三角形的函数:bool IsIsosceles ()
    判断是否是等边三角形的函数:bool IsEquilateral ()
    判断是否是直角三角形的函数:bool IsRightangled ()
    判断是否是钝角三角形的函数:bool IsObtuse ()
    判断是否是锐角三角形的函数:bool IsAcute ()
    析构函数:~Triangle ()
};
```

主函数的格式如下:

```
int main()
{
    定义三角形类的一个对象
```

```
    判断三个边是否构成三角形
    计算三角形面积并显示

    输入三个实数
    设置三角形的三边
    判断三个边是否构成三角形
    计算三角形面积并显示
    判断是否是等腰三角形
    判断是否是等边三角形
    判断是否是直角三角形
    判断是否是钝角三角形
    判断是否是锐角三角形

    return 0 ;
}
```

**【问题扩展】** 使用两个边和夹角重新定义以上的三角形类。

### 2.7.3 日期类的设计与使用

定义日期类，能表示年、月、日，能判断当年是否闰年，显示当月的月历，计算当前日期是星期几，能计算和另一个日期相差的天数等。

月历的显示格式样本如下：

```
MON  TUE  WED  THU  FIR  SAT  SUN
                           1    2
 4    5    6    7    8     9   10
11   12   13   14   15    16   17
18   19   20   21   22    23   24
25   26   27   28
```

**【问题分析】** 定义一个日期类，包括年、月、日三个私有成员变量和公有构造函数、析构函数，并增加计算今年是否闰年，显示本月的月历，计算今天是星期几和计算两个日期相差天数的公有成员函数，最后在主函数中定义日期类的一个对象并调用这些函数，显示计算结果。

本题计算“当天是星期几”。已知 1900 年 1 月 1 日是星期一，计算从 1900 年 1 月 1 日到当天的天数 days，除 7 求余，就是星期几，0 是星期天。显示月历也用到这一方法，要计算当月 1 号是星期几。计算两个日期相差的天数，就是两个 1900.1.1 以来的天数相减。

计算 1900.1.1 以来的天数，要计算到当年经过的闰年数，使用的规则是“四年一闰，百年不闰，四百年再闰”。

**【本例目的】** 练习类、成员及对象的基本定义和使用。

**【算法描述】**

定义日期类

定义年、月、日三个私有成员变量；

定义带有表示年、月、日三个参数的公有构造函数；

定义判断当年是否闰年的公有函数；

定义显示本月的月历的公有函数；

定义计算今天是星期几的公有函数；

定义计算两个日期相差天数的公有函数；

定义公有析构函数；

在主函数中定义的两个对象，并初始化，计算今年是否闰年，显示本月的月历，计算今天是星期几，计算两个日期相差天数，并显示。

**【编程指导】** 日期类的基本格式如下：

```
class Date
{
private:
  年、月、日三个成员变量
public:
  构造函数：Date(int year, int month, int day)
  计算今年是否闰年的函数：bool isLeap()
  显示本月的月历的函数：showCalendar()
  计算今天是星期几的函数：int weekday()
  计算两个日期相差天数：int days(Date date2)
  析构函数：~Date()
};
```

主函数的格式如下：

```
int main()
{
    定义两个日期类的对象
    计算今年是否闰年并显示
    显示本月的月历
    计算今天是星期几并显示
    计算两个日期相差天数并显示

    return 0 ;
}
```

**【问题扩展】** 对以上日期类增加显示年历、两个日期(一个绝对日期、一个相对日期)相加的函数。

## 2.7.4 用类实现学生信息统计

学生信息包括学号、姓名以及三门课的分数，班级信息包括班号、班名、学生人数和学生信息，统计学生的平均分，对班级学生(不超过 100 个)按平均分排序等。

**【问题分析】** 定义一个学生信息类，包括学号、姓名、三门课成绩、平均分等 6 个私有成员变量和公有构造函数、析构函数，并增加显示变量值的公有成员函数；定义一个班级信息类，包括班号、班名、学生人数和学生信息类的对象数组等私有成员变量和构造函数、析构函数、输入学生信息的函数、计算班级平均分的函数、按成绩排序的函数和显示班级信息的公有成员函数。在主函数中定义班级类的对象，调用成员函数，显示结果。

**【本例目的】** 练习类、成员、嵌套对象成员及对象、多文件工程的使用。

**【算法描述】**

学生信息类

定义学号、姓名、三门课成绩、平均分等四个私有成员变量；

定义带有学号、姓名、三门课成绩等参数的构造函数，自动计算平均分；

定义设置学生信息的函数；

定义学生信息类的显示成员变量值的函数；

班级信息类

定义班号、班名、学生人数和学生信息类的对象数组等私有成员变量；

定义带有班号、班名等参数的构造函数，初始时班级人数为 0；

定义输入学生信息的函数，每增加一个人的信息，班级人数加 1；

定义计算班级平均分的函数；

定义按成绩排序的函数；

定义显示班级信息的公有成员函数，包括学生信息。

在主函数中定义学生信息类和班级信息类的各一个对象，展示班级类的功能。

**【编程指导】** **学生信息类**定义在一个单独的程序文件(Student.cpp)中，类的基本格式如下：

```
class Student
{
private:
    学号、姓名、三门课成绩等成员变量
public:
    构造函数：Student(int std_no,string std_name,float score[])
    设置学生信息：void set(int std_no,string std_name,float score[])
    显示变量值的函数：void show()
    析构函数：~Student()
};
```

**班级类**定义在一个单独的程序文件(Class.cpp)中，类的基本格式如下：

```
class Class
{
private:
    班号、班名、人数和学生信息数组等成员变量
public:
    构造函数：Class(int class_no,string class_name)
```

```
    析构函数：~Class ()
    输入学生信息的函数:void input()
    求各科平均分并显示的函数：void average();
    按平均成绩排序的函数：void sort();
    显示变量值的函数：void show()
};
```

主函数定义在一个单独的程序文件(Main.cpp,前两个文件也是工程的一部分)中，主函数的格式如下：

```
int main()
{
    定义班级类的对象并初始化
    通过成员函数输入学生信息
    通过成员函数按平均成绩排序
    显示班级信息
    显示班级平均成绩

    return 0;
}
```

**【问题扩展】** 如果要求班级类中,表示学生信息的数组要动态申请,如何实现?

# 2.8 实验8 取其精华 发挥优势——继承

**实验目的**

(1) 掌握类的公有继承、保护继承和私有继承。
(2) 掌握类继承中的构造函数与析构函数的定义和调用顺序。
(3) 掌握类继承中的成员变量与成员函数的变化。

## 2.8.1 黑白点类和彩色点类

定义一个空间中的黑白点类作为基类,包括 x,y,z 三个坐标值和黑白灰度值等私有成员变量以及公有构造函数、析构函数,并增加显示变量值的公有成员函数。定义一个空间中的彩色点类作为派生类,包括红、绿、蓝等三个颜色值的私有成员变量和公有构造函数、析构函数,并增加显示变量值的公有成员函数;最后在主函数中定义黑白点类和彩色点类的各一个对象并调用这些函数,显示结果。

**【本例目的】** 练习类的公有继承方式,以及基类和派生类之间的调用关系。

**【算法描述】**

黑白点类

定义三个坐标值和黑白灰度值等四个私有成员变量;
定义带有四个参数的构造函数;
定义显示变量值的函数;

定义析构函数；

彩色点类，从黑白点类派生出来

新增表示红、绿、蓝颜色值的三个私有成员变量；

定义带有七个参数的构造函数，要初始化基类成员；

定义显示变量值的函数；

定义析构函数；

在主函数中定义两个对象，并显示结果。

**【编程指导】** 黑白点类的基本格式如下：

```
class Point
{
private:
    三个坐标值和黑白灰度值成员变量
public:
    构造函数：Point(int x, int y, int z  , int lighten);
    析构函数：~Point();
    显示变量值的函数：void show();
};
```

彩色点类的基本格式如下：

```
class ColoredPoint:public Point
{
private:
    红、绿、蓝颜色值等三个成员变量
public:
  构造函数：ColoredPoint(int x, int y, int z, int lighten, int red, int green, int blue): Point(x, y,
            z, lighten);
  显示变量值的函数：void show();
  析构函数：~ColoredPoint();
};
```

主函数的格式如下：

```
int main()
{
    定义两个类的对象各一个
    显示结果

    return 0 ;
}
```

**【问题扩展】** 请用类的继承来描述一维点、二维点和三维点之间的继承关系。

### 2.8.2 使用类的继承编写管理公民信息和大学生信息的程序

定义一个公民类作为基类，包括姓名、性别、年龄等三个私有成员变量和公有构造函

数,并增加显示变量值的公有成员函数;定义一个学生类作为派生类,包括学号和班级两个私有成员变量和公有构造函数,并增加显示变量值的公有成员函数;最后在主函数中定义公民类和学生类的各一个对象并调用这些函数,显示结果。

**【本例目的】** 练习类的私有继承方式,以及基类和派生类之间的调用关系。

**【算法描述】**

公民类

　　定义姓名、性别、年龄等三个私有成员变量;

　　定义带有姓名、性别、年龄等三个参数的构造函数;

　　定义析构函数;

　　定义显示变量值的函数;

学生类,从公民类派生出来

　　新增学号和班级两个私有成员变量;

　　定义带有姓名、性别、年龄、学号和班级五个参数的构造函数;

　　定义析构函数;

　　定义显示变量值的函数;

在主函数中定义两个类的对象,并显示结果。

**【编程指导】** 公民类的基本格式如下:

```
class Citizen
{
private:
    姓名、性别、年龄三个私有成员变量
public:
    构造函数:Citizen(string name, char sex, int age)
    析构函数:~Citizen()
    显示变量值的函数:void show()
};
```

学生类的基本格式如下:

```
class Student:private Citizen
{
private:
    学号和班级两个成员变量
public:
    构造函数:Student(string name, char sex, int age, int std_id, string className):
    Citizen(name, sex, age)
    析构函数:~Student ()
    显示变量值的函数:void show()
};
```

主函数的格式如下:

```
int main()
```

```
{
    定义两个类的各一个对象
    显示结果

    return 0 ;
}
```

**【问题扩展】** 定义职员类和教师类均作为公民类的派生类。

## 2.8.3　使用类的继承编写日期时间管理程序

定义一个日期类作为基类，包括年、月、日三个私有成员变量和公有构造函数、析构函数，并增加显示变量值的公有成员函数；定义一个时间类作为派生类，包括时、分、秒三个私有成员变量和公有构造函数、析构函数，并增加显示变量值的公有成员函数；最后在主函数中定义日期类和时间类的各一个对象并调用这些函数，显示结果。

**【本例目的】** 练习类的保护继承方式，以及基类和派生类之间的调用关系。

**【算法描述】**

日期类
　　定义年、月、日三个私有成员变量；
　　定义带有三个参数的构造函数；
　　定义析构函数；
　　定义显示变量值的函数；
时间类，从日期类派生出来，使用保护继承
　　新增表示时、分、秒的三个私有成员变量；
　　定义带有六个参数的构造函数，初始化基类成员；
　　定义析构函数；
　　定义显示变量值的函数；
在主函数中定义两个对象，并显示结果。

**【编程指导】** 日期类的基本格式如下：

```
class Date
{
private:
    年、月、日三个成员变量
public:
    构造函数：Date(int year, int month, int day)
    析构函数：~Date()
    显示变量值的函数：void show()
};
```

时间类的基本格式如下：

```
class Time:protected Date
{
```

```
private:
    时、分、秒三个成员变量
public:
    构造函数：Time(int year, int month, int day, int hour, int minute, int second):Date(year, month,
    day)
    析构函数：~Time ()
    显示变量值的函数：void show()
};
```

主函数的格式如下：

```
int main()
{
    定义两个类各一个对象
    显示结果

    return 0 ;
}
```

**【问题扩展】** 重新设计以上程序，将时间类作为基类，日期类作为派生类。

## 2.9 实验9 统一接口 多种实现——多态

**实验目的**

（1）掌握虚函数的定义和使用方法。
（2）掌握纯虚函数和抽象类的定义和使用方法。
（3）掌握运算符重载函数的定义方法。

### 2.9.1 显示不同形状的字符图形，包括矩形、三角形和菱形等

使用类的多态显示不同形状的字符图形，包括矩形、三角形和菱形等。定义一个字符窗口类作为基类，包括显示变量值的公有成员虚函数、构造函数和析构函数；定义一个矩形类作为派生类，包括显示字符形状图的公有成员函数；定义一个三角形类作为派生类，包括显示字符形状图的公有成员函数；定义一个菱形类作为派生类，包括显示字符形状图的公有成员函数；最后在主函数中定义它们的各一个对象，声明基类的指针，使指针依次指向各派生类对象，通过指针访问派生类的成员函数，显示结果。

**【本例目的】** 练习类的多态、虚函数的使用。

**【问题分析】** 本例练习虚函数实现多态。首先各类中显示图形的成员函数的格式要完全相同，在基类中将显示图形的函数设置为 virtual；其次，在主函数中要声明基类的指针，使该指针依次指向各派生类的对象，通过指针能访问到派生类的成员。

**【编程指导】** 字符窗口类的基本格式如下：

```
class Window
```

```
{
  public:
      构造函数：Window()
      析构函数：~Window()
      显示窗口字符形状图的虚函数：virtual void show()
};
```

矩形类的基本格式如下：

```
class Rectangle:public Window
{
  public:
      构造函数：Rectangle()
      析构函数：~Rectangle()
      显示矩形字符形状图的函数：void show()
};
```

三角形类的基本格式如下：

```
class Triangle:public Window
{
  public:
      构造函数：Triangle()
      析构函数：~Triangle()
      显示三角形字符形状图的函数：void show()
};
```

菱形类的基本格式如下：

```
class Diamond:public Window
{
  public:
      构造函数：Diamond ()
      析构函数：~Diamond ()
      显示菱形字符形状图的函数：void show()
};
```

主函数的格式如下：

```
int main()
{
    定义四个对象,定义基类的指针 p
    使 p 依次指向各派生类对象,通过 p 访问派生类的成员函数 show()
    return 0 ;
}
```

**【测试指南】** 输出结果样本如下：

…………………

```
.             .
.             .
.             .
.             .
.....................

**************
**************
**************
**************
**************
**************

    **
    ****
   *******
  *********
************
**************

    **
    ****
   *******
  *********
************
**************
***********
  ********
   ******
    ****
    **
```

**【问题扩展】**

(1) 如果 Window 类的 show()函数前不加 virtual,结果如何?

(2) 增加基类的其他派生类,比如梯形、圆、多边形等。

### 2.9.2 使用继承定义一组形状类

基类为表示形状的抽象类,派生类有:矩形类、圆类和三角形类,它们有格式相同的计算面积的函数。

定义一个形状抽象类作为基类,包括计算面积的公有成员纯虚函数,构造函数和析构函数;定义一个矩形类作为派生类,包括计算面积的公有成员函数;定义一个圆类作为派生类,包括计算面积的公有成员函数,构造函数和析构函数;定义一个三角形类作为派生类,包括计算面积的公有成员函数,构造函数和析构函数;最后在主函数中定义它们的各

个对象并调用这些函数，显示结果。

**【本例目的】** 练习类的多态、纯虚函数和抽象类的使用。

**【问题分析】** 抽象类就是至少有一个纯虚函数的类。纯虚函数在函数声明的最前面加 virtual，在后面写“＝0”，没有定义。抽象类不能用来声明对象，只用来继承，作用就是给出格式一致的成员函数。对本题来说，形状类的作用就是规定求面积的函数的格式，不管是什么形状，求面积的函数的格式都相同。

**【编程指导】** 形状抽象类的基本格式如下：

```
class Shape
{
  public:
      构造函数：Shape()
      析构函数：~Shape()
      计算面积的纯虚函数：virtual double Area()=NULL;   //0与 NULL一样
};
```

矩形类的基本格式如下：

```
class Rectangle:public Shape
{
private:
  成员变量：宽和高
public:
  构造函数：Rectangle()
  构造函数：Rectangle(int width,int height)
  析构函数：~Rectangle()
  计算面积的函数：double Area()
};
```

圆类的基本格式如下：

```
class Circle:public Shape
{
  private:
    成员变量：半径
  public:
    构造函数：Circle()
    构造函数：Circle(int radius)
    析构函数：~Circle()
    计算面积的函数：double Area()
};
```

三角形类的基本格式如下：

```
class Triangle: public Shape
{
```

```
  private:
    成员变量：三个边长
  public:
    构造函数：Triangle()
    构造函数：Triangle(int a,int b,int c)
    析构函数：~Triangle()
    计算面积的函数：double Area()
};
```

主函数的格式如下：

```
int main()
{
    定义四个对象
    计算面积并显示结果

    return 0 ;
}
```

**【问题扩展】**

(1) 有纯虚函数的 Shape 类可以声明对象吗?

(2) 声明基类的指针,通过指针访问各派生类对象的成员。

(3) 再增加基类的其他派生类,如梯形、平行四边形等。

### 2.9.3 重载运算符实现复数类的四则运算

定义一个复数类,私有数据成员包括实部和虚部,重载运算符+,-,*,/实现复数的四则运算,添加其他需要的成员函数。编写主函数,展示复数类的功能。

**【本例目的】** 练习类中的运算符重载函数的定义和调用。

**【编程指导】** 复数类的基本格式如下：

```
class Complex
{
private:
    实部和虚部成员变量
public:
    构造函数：Complex(double real=0,double imag=0) ;
    析构函数：~Complex()
    加法运算符重载函数：Complex operator+ (const Complex &c2)
    减法运算符重载函数：Complex operator- (const Complex &c2)
    乘法运算符重载函数：Complex operator* (const Complex &c2)
    除法运算符重载函数：Complex operator/(const Complex &c2)
    显示函数：void Show()
};
```

# 2.10 实验 10 文件与输入输出

**实验目的**

(1) 熟悉常见的输入输出方式。
(2) 理解不同类型文件的差异。
(3) 掌握文件打开、关闭的方法。
(4) 掌握文本文件的基本读写方法。
(5) 掌握二进制文件的基本读写方法。
(6) 熟悉随机文件的基本读写方法。

## 2.10.1 格式化输出数据

将整数 1～25 的三次方分 5 行输出,每个数字都右对齐输出在域宽为 6 的范围内。再将实数 3.1415926 分别取 1～5 位小数分 5 行输出。其显示效果如下:

```
     1     8    27    64   125
   216   343   512   729  1000
  1331  1728  2197  2744  3375
  4096  4913  5832  6859  8000
  9261 10648 12167 13824 15625
3.1
3.14
3.142
3.1416
3.14159
```

**【本题目的】** 练习格式化输出中的几个基本方法。

**【编程提示】** 设置域宽用 setw()函数或 cout 的 width()函数设置。设置精度则可使用 setprecision()函数。另外,为了将 25 个数字分 5 行输出,可以采用处理矩阵常用的两重循环嵌套结构。注意某些函数可能会用到头文件 iomanip.h(或 iomanip)。

**【问题扩展】** 尝试将圆周率按科学计数法输出,尝试将域内空的部分用“ * ”填充。

## 2.10.2 文件中特定单词的统计

在日常工作中,常常需要统计一些特定单词的使用次数。本题目就来做以下尝试:从文本文件(文件名 text.txt)中读取一篇英文短文的内容,统计其中单词“are”的使用次数并在屏幕上输出。文本文件请用记事本自行创建。

注意,单词“are”一定不在句首,因此它的前一个字符一定是空格(因为这个单词如果在句首应写作“Are”)。但是这个单词的后一个字符不一定是空格,可能是逗号、句号等。请创建文本时注意各类情形都应出现。

**【本题目的】** 练习基本的文本文件打开、读取方法。练习字符串的判别方法。

**【问题分析】** 解决问题的关键是文件打开后，如何判断读到单词“are”。一种方法是考察每一个位置开始的连续5个字符，若依次是' '、'a'、'r'、'e'和标点符号（或空格），则找到一个单词。

首先，创建ifstream类的对象，使用其打开文件后，按字节流方式逐字节读取文件内容，为后面的判断做好准备。同时，应增加判断语句判断文件打开成功与否。

其次，根据本题目特点，可以边读边作统计处理。如果读到空格，则再读取4个字符，看看是不是“are”加上一个非字母字符（即符号）。如果是，就记入统计次数。

最后，输出信息，关闭文件。

**【算法描述】**

```
定义ifstream对象，并使用其打开文件；
连续读取后4个字符到字符数组str[0]、str[1]、str[2]、str[3]中；
当文件没有读完{
    读取1个字符到str[4]；
    如果str的前4个字符依次是' '(空格)、'a'、'r'、'e'，则
    {
        若str[4]不是字母，则
        {    统计数字加1；  }                          //找到一个
    }
    将str[1]至str[4]依次赋值给str[0]至str[3]；          //前移一个位置
}
输出统计次数；
```

**【测试指南】** 可以分别测试单词are后面是字母、空格、标点等情况，看看程序能否正确执行。

**【问题扩展】**

（1）如果不使用每次向后移动一个字符的方法，有没有其他方法可以一次读取一个单词？如果有这样的方法，你认为有什么优点和缺点？能否编程序实现你的想法？

（2）如果不考虑大小写，那么ARE也算一个单词了。若想统计这样的单词，如何修改程序？

## 2.10.3 分离文本文件中的英文和中文

有一个文本文件，每一行都是一个单词或词组及其中文解释，英文和中文之间可以无空格。下面是一个样例：

```
China   中国
Foreign Minister外交部长
Sino-Russian tie   中俄关系
last year去年
```

编写程序,将上面这样的文件中的英文写入文件 out1.txt,将中文写入文件 out2.txt。都是每行一个词汇。

**【本题目的】** 练习基本的文本文件打开、读取、写入方法,以及同时操作多个文件的方法。练习字符的判别方法。

**【问题分析】** 除了打开、关闭、读取、写入文件之外,主要要将每一行中的中文识别出来。每个中文字都是有两个字节构成,这两个字节转换成整数都小于0(即最高位为1)。利用这一点就可以识别中文。

下面算法里中文的识别方法是:将一个字节读入字符变量,然后将该变量转换为整数。如果该整数大于0,就说明这个字节是英文;若小于0,则这个字节是中文的一部分。

根据本题目的情况,可以边读文件边处理。这样程序对内存的需要将很小,也提高了执行效率。

注意,打开文件时,应增加判断语句判断文件打开成功与否。

**【算法描述】**

```
定义 ifstream 对象,并使用其打开文件;
定义 ofstream 对象,并使用其打开文件 out1.txt;
定义 ofstream 对象,并使用其打开文件 out2.txt;
当(文件没有读完){
    读取一行内容到字符数组 str;
    计数器 i=0;
    当(str[i]!='\0' 且 (int)str[i] > 0 )    //不是中文
    {
        字符写入 out1.txt;
    }
    输出换行符到 out1.txt
    将剩余 str 数组内容写入 out2.txt;
    输出换行符到 out2.txt
}
关闭所有文件;
```

**【结果示例】**

英文文件的内容为:

```
China
Foreign Minister
Sino-Russian tie
last year
```

中文文件的内容为:

```
中国
外交部长
```

中俄关系
去年

**【问题扩展】** 如果不小心文件的某一行出现了两个英文单词及其中文解释放在一行的情况，即形如：

"China 中国 Foreign Minister 外交部长"

的情形。那么程序还能正常运行吗？若不能，有没有改进的方法？

## 2.10.4 有格式文本文件的创建及读取

从键盘读入若干行数据（行数小于 100），每行的数据依次为英文单词、中文解释、单词出现次数，这些数据以空格隔开。最后一行输入“－1 －1 －1”表示输入结束。将输入的数据逐行写入文件 fa.txt（最后一行三个－1 不要写入文件）。然后再将文件打开，读取英文单词和单词出现次数，并按照单词出现次数由高到低逐行输出在屏幕上。

**【本题目的】** 练习文件打开、关闭的方法，有格式文本文件的基本读写方法。

**【问题分析】** 首先看写文件的阶段。英文单词和中文解释应该用字符数组表示，而单词出现次数则可用整型变量表示。屏幕输入的信息可以按行读取并逐行写入文件。写入数据过程用符号“＜＜”即可。写入完成一定要关闭文件，以便下一步操作。

再看读文件的阶段。由于要读入多个英文单词，所以可以用二维数组存储这些单词。而所有使用次数则用一维整型数组即可。比如，可以用 word[100][20]、num[100]存储 100 个单词的内容和使用次数。word[k][20]和 num[k]代表同一行中的两个数值。

排序过程需要对 num 数组进行，注意是由大到小排序。在对 num 数组的内容操作时，比如交换 num[i]和 num[j]，则需要同时交换 word[i][20]和 word[j][20]，以保持单词和使用次数的对应。

**【算法描述】**

```
定义 ofstream 对象 fa.txt；
定义数组 w[20]、c[20]、n；
当(w[20]、c[20]和 n 不是－1)
{
    从屏幕读入 w[20]、c[20]、n；
    若(w[20]、c[20]和 n 不是－1)
        向文件写入 w[20]、c[20]、n；
}
关闭文件；
再次打开文件；
定义 word[100][20]、num[100]；
从文件读取所有单词及其使用次数到 word、num 数组；
对数组排序；
```

输出 word、num 数组；

**【问题扩展】** 用二维数组解决本问题是传统 C 语言的解决方式，还可以用 string 数组来存储多个字符串，那样更方便一些，请尝试实现一下。

## 2.10.5　学生成绩信息的处理

已知一个文件内容是学生成绩信息。每一行的内容依次是学号、姓名、性别、高数成绩、物理成绩、英语成绩。样例如下：

```
010010    张三    男    87    90    86
010011    李四    女    79    95    90
……        ……      ……    ……
```

编写程序，读取这样的文件（假定已知人数小于 100，数量未知），将所有女生的成绩信息在屏幕上逐行输出，最后再输出所有女生三门课的平均成绩。

本题目文本文件请自行建立。

**【本题目的】** 练习读取特定格式的文本文件的方法，练习结构体的使用。

**【问题分析】** 读入文件时，由于要读入多个字符串（如学号、姓名、性别），当然可以用二维数组存储这些串。但这里采用其他方法来处理，将学生的信息存为结构体的形式。其定义方式可采用下面的形式。

```
struct STUDENT {
    char ID[10];        //学号
    char Name[15];      //姓名
    char Sex[3];        //性别
    int  math;          //高数成绩
    int  phy;           //物理成绩
    int  eng;           //英语成绩
};
```

由于要计算所有女生的平均分，因此可以定义一个 STUDENT 数组，将所有女生的信息保存到数组中，最后计算平均分。另外可以定义三个变量 SumMath、SumPhy、SumEng 用于累加所有女生各科的成绩，累加过程可以在读数据时进行，同时记录女生的人数，最后就可以算出平均分。

**【算法描述】**

定义并初始化 count 用于统计人数；

定义并初始化 SumMath、SumPhy、SumEng；

STUDENT stu；

利用 ifstream 打开

当（文件尚未读完）

{

　　利用“>>”符号读取一行信息到 stu；

　　若（stu. Sex 为“女”）{

```
            输出信息；
            累加各科的成绩到 SumMath、SumPhy、SumEng；
            count ++；
        }
    }
    计算平均分并输出到屏幕；
```

**【思考问题】** 什么时候必须用数组先将学生的信息全保存下来，再作进一步处理。试举出几个例子。

## 2.10.6 读取 BMP 文件的宽度和高度

BMP 文件是典型的二进制文件，其文件头部存储的内容是文件信息区和图像信息区。前 14 个字节为文件信息区，包括文件类型标识、文件大小等。之后 40 个字节是图像信息头，其中前三个信息包括位图信息头的大小(字节数，占 4 字节)，位图宽度(像素数，占 4 字节)和位图高度(像素数，占 4 字节)，其他内容请感兴趣的读者自己查阅资料。

本题目要求读取一个 BMP 文件的宽度和高度。

**【本题目的】** 练习以二进制形式打开文件，然后定位内容并读取信息。

**【问题分析】** 位图文件(BMP 图像文件)是二进制文件。要读取指定的字节，首先要以二进制方式打开，即在打开文件时，打开方式使用“ios::binary”。字节的定位使用输入流的成员函数 seekg(偏移量，参照位置)，从头开始定位，参照位置写“ios::beg”，宽度的偏移量是 14+4，高度的偏移量是 14+4+4。

**【编程提示】** 根据上文提到的 BMP 文件结构，读取长、宽只要从文件头部偏移 18 字节的位置开始读两个整数就行。

**【问题扩展】** 如何判断读取的文件的确是 BMP 文件呢？请查阅 BMP 文件的说明，提出解决方法并加以实现。

## 2.10.7 用随机文件存储书籍信息

书籍信息包括书名、作者、单价三部分，其中书名和作者都是用长度为 60 的 char 类型数组表示，单价用 double 类型变量表示。请建立随机文件 fa.dat，并依次存入如下四本书的信息：

```
C++程序设计          王刚        70.5
MySql 宝典           刘大同      80.5
STL 全面解析         程力        68.0
Sql Server 详解      周汉        58.0
```

关闭文件后再次打开，读取并在屏幕上输出第 2 本和第 4 本书的信息。

**【本题目的】** 练习随机文件的建立及其读写操作。

**【问题分析】** 首先，可以建立以下结构体存储书籍信息。

```
struct Book {
```

```
        char title[60];             //书名
        char author[60];            //作者
        double price;               //单价
    };
```

其次，为读取第 2 本和第 4 本书的信息，需要在文件中定位。这可能要用到 seekp 或 seekg 函数。seekp 用于 ofstream 对象，seekg 用于 ifstream 对象。

采用 ofstream 对象建立文件并写入信息，再利用 ifstream 对象打开文件并读取信息。注意，打开文件前要先用 ofstream 对象的成员函数 close()关闭文件。

**【算法描述】**

① 定义 Book 结构体数组并初始化。

② 利用 ofstream 创建对象并建立(打开)文件。

③ 循环写入结构体数组信息到文件。

④ 关闭文件。

⑤ 利用 ifstream 打开文件。

⑥ 在文件中定位第 2 本书的位置。

⑦ 读取信息，输出信息。

⑧ 在文件中定位第 4 本书的位置。

⑨ 读取信息，输出信息。

⑩ 关闭文件。

**【测试指南】** 尝试一下，如果定位第 2 本书的位置出现偏差，会读出什么信息。

**【问题扩展】** 有没有办法只打开一次文件不仅进行读操作，还可进行写操作？fstream 类可以做到这一点。请读者参阅 fstream 类的说明，尝试改造一下程序。

## 2.11　实验 11　数据结构与算法

**实验目的**

(1) 了解标准模板库(STL)的作用。

(2) 了解标准模板库的内容。

(3) 熟悉标准模板库所表达的数据结构。

(4) 提高算法的理解和设计的能力。

### 2.11.1　手工操作 Hanoi 塔

本题目要求实现下列功能：假定 Hanoi 塔中盘子自上而下编号为 1，2，3，…，设法存储三层 Hanoi 塔的三个柱 A、B、C 的状态。然后以手工方式输入盘子的移动过程，比如输入“A　B”表示将 A 上的盘子移动到 B 上。每输入一次，就显示一次三个柱的状态，直到问题解决(注：Hanoi 塔的问题描述及移动规则参见主教材递归部分的讲解)。

图 2-3 中层 Hannoi 塔的状态可写作“ A：21　　B：　　C：”，它表示 A 柱自下而上的盘子是 2 号和 1 号盘，B、C 上均无盘。

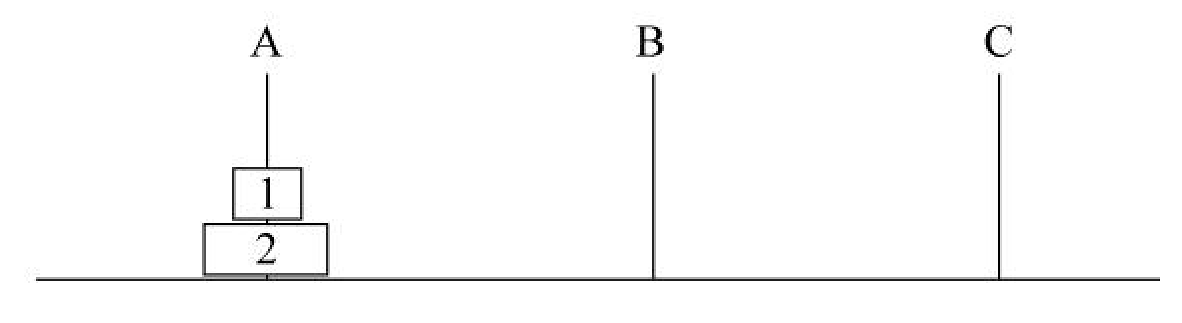


图 2-3　一个两层 Hanoi 塔初始状态

以下是针对两层 Hanoi 塔的执行过程：

```
Hanoi 塔状态：A:21      B:       C:         输入移动操作：A  B
Hanoi 塔状态：A:2       B:1      C:         输入移动操作：A  B
移动操作无效！        输入移动操作：A  C
Hanoi 塔状态：A:        B:1      C:2        输入移动操作：B  C
Hanoi 塔状态：A:        B:       C:21       成功！
```

**【本题目的】** 熟悉堆栈的概念，练习运用 vector 模拟堆栈的方法。

**【问题分析】** 该问题不需要找到解决 Hanoi 塔问题的步骤，只是模拟 Hanoi 塔的状态。由于三个柱 A、B、C 都遵循数据先进后出的原则，因此用堆栈结构进行模拟就可以了。因为 stack 结构无法读取其每一个元素的值，可以用标准模板库里的 vector 来做模拟。

由于有三个柱，因此可定义三个 vector 对象如下：

```
vector<int> v[3];                         //v[0]---A柱   v[1]---B柱   v[2]---C柱
for(int i=0; i<3; i++)    v[i].reserve(3);        //每个容量都为 3
```

假定由于 A 柱有三个盘，由大到小用 3、2、1 表示，则 v[0]初始化如下：

```
for(int i=0; i<3; i++)  {  v[0].push_back(3-i);  }
```

若要显示 A 柱的内容，可采用下列语句：

```
for(int i=0; i<v[0].size(); i++)          //v[0].size()为实际元素数量
    cout<<v[0].at(i);
```

若要得到 A 柱最后一个元素，可采用下列语句：

```
v[0].back();                              //注意：本语句不判断 A柱是否为空
```

若要删除 A 柱最后一个元素，可采用下列语句：

```
v[0].pop_back();
```

注意，某些函数(例如 back)不判断 vector 是否为空，若直接执行有可能出错，因此可先进行相关判断再执行这些函数。

对于 3 层 Hanoi 塔，给出其移动过程及状态变化如下，以便参考。

| 移动 | A | B | C |
|---|---|---|---|
| | 321 | 0 | 0 |

```
执行 A->C后          32      0       1
执行 A->B后          3       2       1
执行 C->B后          3       21      0
执行 A->C后          0       21      3
执行 B->A后          1       2       3
执行 B->C后          1       0       32
执行 A->C后          0       0       321
```

**【算法描述】**

① 定义并初始化 3 个 vector 对象。

② 若 A 柱、B 柱对应的 vector 为空，则成功，结束程序，否则执行下面的步骤。

③ 读取用户输入的两个字符，并转化为对应 vector 的下标。

④ 读取要执行出栈操作的 vector 对象的最后一个元素，记作 out。

⑤ 读取要执行入栈操作的 vector 对象的最后一个元素，记作 in。

⑥ 若 out > in，则不符合规则，转回③。

⑦ 执行出栈、入栈操作，转回②。

**【问题扩展】** 这里循环输出 vector 内容时，采用了计算 vector 长度的函数 size。实际上，更好的做法是利用 vector 的 begin 函数传回迭代器中的第一个数据地址，用 end 函数得到迭代器中末端元素的下一个（指向一个不存在元素）位置的地址，而后用迭代器 iterator 进行数据遍历。下面是一段示例：

```
vector< int> ::iterator p;          //定义迭代器
p=myVec.begin();                    //指向容器的首个元素
p++;                                //移动到下一个元素
*p=20;                              //则 myVec 容器中的第二个值被修改为 20
//循环扫描迭代器,改变所有的值
for(p=myVec.begin();  p!=myVec.end();  p++)
{        *p=50;        }
```

显然，迭代器的用法与指向数组元素的指针很相似。

## 2.11.2　模拟有限长队列

假设内存中有一块长度为 5 的整型空间，它为两个程序提供通信的渠道。一个程序向这块空间依次存入数据（生产者），另一个程序从这块空间依次读取并删除数据（消费者），数据排队进入这块空间，并按先进先出的顺序消费数据。

请编程模拟这一过程用户输入“i　5”表示生产了一个数据 5 放入队列；用户输入“o”表示消费了一个队列头部的数据。若生产数据时，空间满，则返回“空间满，数据丢弃”；若消费数据时，空间是空的，则返回“没有数据，请等待！”。输入“x”则程序结束。每一次操作后，都显示一次队列的状态。典型的执行过程如下：

```
i  5
队列状态：5
```

```
i 9
队列状态：5 9
i 1
队列状态：5 9 1
o
队列状态：9 1
o
队列状态：1
o
队列状态：
o
没有数据,请等待!
x
```

**【本题目的】** 了解队列结构的操作,练习利用 vector 模拟队列的方法。

**【问题分析】** 该问题的核心是模拟一个长度为 5 的队列,可以用 vector 实现。

入队操作可采用 push_back 函数,从尾部插入新数据。

出队操作可采用 erase 函数实现,例如名为 v 的 vector 对象在删除队列头部元素时采用 v.erase(v.begin())即可。

下列这段程序,向 vector 对象增加元素 0~9,而后删除队列头部元素。可作为参考。

```
#include <iostream>
#include <vector>
using namespace std;
int main()
{
    vector<int> v;
    int i;
    for (i=0; i<10; i++)    {    v.push_back(i);   }
    //显示队列头和队列尾
    cout<<"队头："<<v.front()<<"  队尾："<<v.back()<<endl;
    //显示队列中的内容,显示 0 1 2 3 4 5 6 7 8 9
    for(int i=0;i<v.size();i++)     cout<<v.at(i)<<" ";
    v.erase(v.begin());             //删除队列头部元素
    cout<<'\n';
    //再次显示队列中的内容,显示 1 2 3 4 5 6 7 8 9
    for(int i=0;i<v.size();i++)    cout<<v.at(i)<<" ";
    system("pause");                //屏幕停顿
    return 0;
}
```

**【算法描述】**

定义 vector 对象；
循环读入数据，若读入数据是字符 x，则退出循环
{
    若读入数据是字符 i，则读取后面整数入队，显示队列状态；
    若读入数据是字符 o，则出队，显示队列状态；
}

**【问题扩展】** 其实在标准模板库中，还有一个对象——queue，这是一个标准的队列。它有入队和出队等标准函数。遗憾的是，它不能访问队列中间的元素，只能访问队头和队尾元素。

## 2.11.3　黑白棋游戏

黑白棋（也称反棋）是一种棋类游戏，可以用围棋棋盘和棋子进行游戏。初始状态在棋盘的中央有如图 2-4 所示的四个棋子。

黑白双方轮流行棋。当黑方下棋时，所下棋子必须至少与另一颗黑棋之间夹住一串成直线或斜线的白子，且两黑子间除了白子不能有其他空位，落子后被黑棋夹住的白棋变为黑棋。白方下棋时，遵守类似的规则。最终棋子多的一方获胜。

图 2-4　黑白棋初始状态

请编写程序，实现这一游戏。显示时以 0 代表空位，以"＊"代表黑棋，以"＋"代表白棋，以 0、＊、＋ 构成的方阵代表棋盘情况。以下是程序运行样例。

```
棋盘状态如下,轮黑方下棋
   0  1  2  3   4  5  6  7
0  0  0  0  0   0  0  0  0
1  0  0  0  0   0  0  0  0
2  0  0  0  +   *  0  0  0
3  0  0  0  *   +  0  0  0
4  0  0  0  0   0  0  0  0
5  0  0  0  0   0  0  0  0
输入：2  2
棋盘状态如下,轮白方下棋
   0  1  2   3   4   5  6  7
0  0  0  0   0   0   0  0  0
1  0  0  0   0   0   0  0  0
2  0  0  *   *   *   0  0  0
3  0  0  0   *   +   0  0  0
4  0  0  0   0   0   0  0  0
5  0  0  0   0   0   0  0  0
输入：1  2
```

```
棋盘状态如下,轮黑方下棋
   0  1  2  3  4  5  6  7
0  0  0  0  0  0  0  0  0
1  0  0  +  0  0  0  0  0
2  0  0  *  +  *  0  0  0
3  0  0  0  *  +  0  0  0
4  0  0  0  0  0  0  0  0
5  0  0  0  0  0  0  0  0
输入:1  3
棋盘状态如下,轮白方下棋
   0  1  2  3  4  5  6  7
0  0  0  0  0  0  0  0  0
1  0  0  +  *  0  0  0  0
2  0  0  *  *  *  0  0  0
3  0  0  0  *  +  0  0  0
4  0  0  0  0  0  0  0  0
5  0  0  0  0  0  0  0  0
输入:3  2
棋盘状态如下,轮黑方下棋
   0  1  2  3  4  5  6  7
0  0  0  0  0  0  0  0  0
1  0  0  +  *  0  0  0  0
2  0  0  +  *  *  0  0  0
3  0  0  +  +  +  0  0  0
4  0  0  0  0  0  0  0  0
5  0  0  0  0  0  0  0  0
……
```

**【本题目的】** 理解黑白棋的存储方法,理解黑白棋的走棋算法,掌握其中数组的定义及循环控制方法,练习基本输出方法。

**【问题分析】** 以二维数组作为棋局的存储结构。这是后续算法实现的基础,二维数组每个元素对应棋盘一个交叉位置。当该位置无棋时,则该元素为0;当前位置为黑棋时,该元素为1;当该位置为白棋时,该元素为−1。注意,二维数组内容并不一定与显示的字符一样,这里设置二维数组内容为整数可以更方便地编程。

下面分析一下每落下一子的判断算法。每落下一子时,需要判断该子在左右上下及±45°和±135°八个方向上能否夹住对方的一串棋子。假如任何一个方向上都不能夹住对方的棋子,则该子下法不合规则,这次走棋无效。若一方无处落子,则由另一方连续下子。下面以伪代码描述落子的算法。

```
//假定 a[][]存储棋盘状况,(i,j)为落子位置
int  go(int  i,int  j,  int  c)                    //c为颜色,取值 1、0或-1
```

```
{
    int  k, p;
    int  num=0;
    //判断自(i,j)向右方向的情况
    k=i+1;
    while(a[k][j]*c==-1)   { k=k+1; }           //遇到对方的棋则继续向前
    if (a[k][j]*c==1且 k≠i+1且 k未超界)
    {   //有连续对方棋子被夹住,将夹住的棋子反色
        for(int m=i+1; m<k; m++)    a[m][j]=c;
        num=num+k-i-1;                          //累加计算反色棋数
    }
    //判断自(i,j)向左方向的情况
    k=i-1;
    while(a[k][i]*c==-1){  k=k-1;}
    if(a[k][j]*c==1且 k≠i-1且 k未超界)
    {   //有连续对方棋子被夹住,将夹住的棋子反色;
        for(int m=i-1; m>k; m--)    a[m][j]=c;
        nun=num+i-k-1;                          //累加计算反色棋数
    }
    //判断自(i,j)向下方向的情况
    k=j+1;
    while(a[i][k]*c==-1) {  k=k+1;}
    if(a[i][k]*c==1且 k≠j-1且 k未超界)
    {   //有连续对方棋子被夹住,将夹住的棋子反色;
        for(int m=j+1; m<k; m++)  a[i][m]=c;
        nun=num+k-j-1;                          //累加计算反色子数
    }
    //判断自(i,j)向上方向的情况
    k=j-1;
    while(a[i][k]*c==-1) {  k=k-1;}
    if(a[i][k]*c==1且 k≠j-1且 k未超界)
    {   //有连续对方棋子被夹住,将夹住的棋子反色;
        for(int m=j-1;m>k; m--)    a[i][m]=c;
        nun=num+j-k-1;                          //累加计算反色子数
    }
    //判断自(i,j) 沿着 45°方向的情况
    k=i+1; p=j-1;
    while(a[k][p]*c==-1) {  k=k+1, p=p-1;}
    if(a[k][p]*c==1且 k≠i-1且 k,p未出界)
    {   int  m=i+1,n=j-1;
        while(m<k)
        {   a[m][n]=c;
            m=m+1; n=n-1;
        }
```

```
        num=num+k-i-1;                              //累加计算反色子数
    }
    //判断自(i,j)沿着 135°方向的情况
    k=i-1; p=j-1
    while(a[k][p]*c==-1) {   k=k-1, p=p-1;}
    if(a[k][p]*c==1 且 k≠i-1 且 k,p 未出界)
    {   int   m=i-1,n=j-1;
        while(m>k)
        {   a[m][n]=c;
            m=m-1, n=n-1;
        }
        num=num+i-k-1;                              //累加计算反色子数
    }
    //判断自(i,j)沿-45°方向的情况
    k=i+1; p=j+1;
    while(a[k][p]*c==-1) {   k=k+1;p=p+1}
    if(a[k][p]*c==1 且 k≠i+1 且 k,p 未出界)
    {   int   m=i+1,n=j+1;
        while(m<k)
        {   a[m][n]=c;
            m=m+1;n=n+1
        }
        num=num+k-i-1;                              //累加计算反色子数
    }
    //判断自(i,j)-135°方向的情况
    k=i-1; p=j+1
    while(a[k][p]*c==-1) {      k=k-1;p=p+1; }
    if(a[k][p]*c==1 且 k≠i-1 且 k,p 未出界)
    {   int   m=i-1,n=j+1;
        while(m>k)
        {   a[m][n]=c;
            m=m-1;n=n+1
        }
        num=num+i-k-1;                              //累加计算反色子数
    }
    return     num;                                 //若 num=0 代表落子不合法
}
```

二维数组 a[][]存储棋盘的状况，在显示时将数组 a 中为 1 的元素显示为“*”，将 a 中为－1 的元素显示为“＋”，其他位置显示为 0 即可。

**【算法描述】**

① 定义二维数组 a[][]，初始化为中心有 4 个棋子。

② 显示棋盘状态。

③ 黑方输入，调用 go 函数，记录落子成功与否。

④ 显示棋盘状态。

⑤ 白方输入,调用 go 函数,记录落子成功与否。

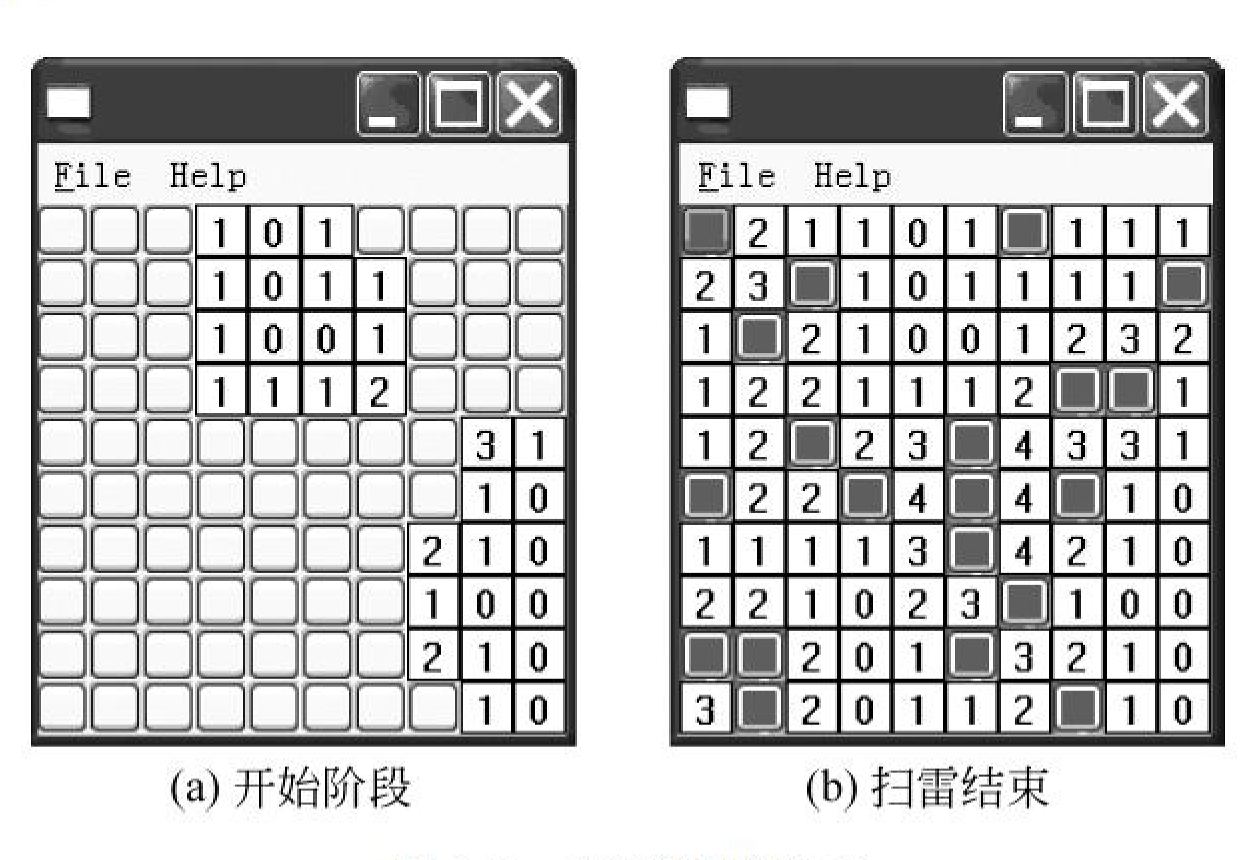
⑥ 若双方至少有一方落子成功,则返回②,否则执行⑦。

⑦ 统计双方棋子数,判断哪方获胜,输出结果。

**【问题扩展】** 注意,上面给出的落子算法 go 函数以将对方棋子反转的数量作为返回值。若这个值大于 0 则走法正确,反之则走法错误。若一方落子不正确,应继续让该方重新走棋。但是,问题在于,如果一方从开始就已经无棋可走,那么是不应该让他走棋的。怎么判断某一方已经无棋可走?这个问题较为复杂,读者可以思考一下。为避免某一方无棋可走时进入死循环,一个简单的解决方法是,让某一方输入一个数字(比如-1)代表放弃走棋,这样游戏仍可继续。

## 2.11.4 生成地雷阵

图 2-5 是一款扫雷游戏,除界面稍有不同,它与 Windows 自带的扫雷游戏完全一样。单击砖块,若是非地雷区就会翻开,并显示一个数字,该数字代表其周围至多八个砖块下有几个地雷。对于玩家认为是地雷的砖块,玩家可以右键单击标记之,扫雷结束后如图 2-5(b)所示。显然,在游戏开始时,地雷的位置和非地雷区域的数字就已经产生了,只不过隐藏在砖块之下。



(a) 开始阶段 (b) 扫雷结束

图 2-5 扫雷游戏界面

本题目就是要编程生成一个 10×10 的二维数组,用来表示类似于上面的隐藏在砖块之下的地雷阵。数组每个单元对应地雷阵的一个砖块。若某个单元位置为地雷,则用-1 表示;若该位置不是地雷,则显示一个数字,该数字为其周围至多八个砖块下的地雷数目。要求地雷数量由用户输入,地雷位置随机产生,最后将生成的二维数组在屏幕上输出。为显示美观,凡是应显示-1(地雷)的位置用字母 m 代表。下面是一个执行样例。

```
请输入地雷数量:20
m 1 0 1 m m 1 0 0 0
1 1 0 2 3 3 2 1 1 0
0 0 0 2 m 3 2 m 1 0
1 2 2 3 m 4 m 2 1 0
```

```
1  m  m  2  2  m  2  2  1  1
2  4  4  3  2  2  2  2  m  2
1  m  m  2  m  1  1  m  4  m
2  3  3  2  1  1  1  2  m  2
2  m  2  0  1  1  1  1  1  1
2  m  2  0  1  m  1  0  0  0
```

**【本题目的】** 练习循环嵌套、二维矩阵处理、随机数生成。

**【问题分析】** 伴随 Windows 几十年成长历程的扫雷游戏从算法角度看并不复杂，主要就是生成地雷矩阵和扫雷过程控制两部分算法。生成地雷矩阵基本可分为以下几步：

① 初始化矩阵。

② 随机生成地雷位置。

③ 修改地雷周围的信息。

由于每个非地雷单元存放的数字代表其周围的地雷数量，而初始时刻没有设置地雷，所以将二维数组的内容全部初始化为 0 是必然选择。

随机生成地雷位置的方法很多。本例将二维数组看作先自左向右、再逐行向下连续编号的单元(从 0 开始)，比如 10×10 的方阵可编号为 0～99。在生成随机数时，也产生一个小于数组单元总数的数字，这样随机数恰好对应于按前面方法编号的单元，该处就是地雷位置。若新生成的地雷位置与原有的地雷位置重复，则再次生成新位置，直到该处可放地雷为止。下面是生成一个地雷位置的算法：

```
do     //反复生成位置,直到该处到可放地雷
{
    生成小于 100 的随机数放入整数 index;
    计算 index 单元对应的行 x 和列 y;
}while(若数组(x, y)处为-1);              //-1 表示已有地雷
设数组(x, y)处为-1;
```

当每次地雷设置完毕后，可以立即将其周边非雷区单元的数字加一。设(x，y)处为地雷，则算法如下：

```
循环(i 从 x-1 到 x+1 且 0≤i≤9)
    循环(j 从 y-1 到 y+1 且 0≤j≤9)
        若(i, j)处不为-1,则该处单元加 1;
```

以上布雷和修改数字过程应循环执行，循环次数与用户输入地雷数一致。

下面给出算法完整的伪代码形式：

```
定义 10×10 矩阵 matrix 并初始化为全 0,生成的地雷数 k=0;
输入地雷数量 mineNum
while(生成的地雷数 k<mineNum)
{
    do                                  //反复生成位置(x,y),直到该处到可放地雷
    {
        生成小于 100 的随机数,将其十位取作行 x,个位取作列 y;
```

```
    }while(若数组(x, y)处为－1);
    设数组(x, y)处为－1;                //设置(x, y)处为地雷
    //修改矩阵 matrix 在(x,y)周围格子的内容
    循环(i 从 x-1 到 x+1 且 0≤i≤9)
        循环(j 从 y-1 到 y+1 且 0≤j≤9)
            若(i, j)处不为－1(地雷),则该处单元加 1
    k++;                               //地雷数加 1
}
循环显示矩阵内容
```

**【程序参考】**

```
#include <iostream>
using namespace std;
int main()
{
    int i,j,mineNum;
    int matrix[10][10];
    for(i=0;i<10;i++)
        for(j=0;j<10;j++)
        {
            matrix[i][j]=0;
        }
    cout<<"请输入地雷数量:";
    cin>>mineNum;
    int k=0;
    while(k<mineNum)
    {
        int index, x, y;
        //反复生成位置,直到该处可放地雷
        do
        {
            index=rand()%(10*10);
            x=index/10,y=index%10;
        }while(matrix[x][y]==-1);
        matrix[x][y]=-1;
        //地雷周围格子的数字加 1
        int a1=x-1>0?x-1:0;
        int a2=x+1<9?x+1:9;
        int b1=y-1>0?y-1:0;
        int b2=y+1<9?y+1:9;
        for(i=a1;i<=a2;i++)
            for(j=b1;j<=b2;j++)
            {
                if(matrix[i][j]!=-1)     matrix[i][j]++;
            }
```

```
            k++;            //地雷数加 1
        }
        for(i=0;i<10;i++)
        {
            for(j=0;j<10;j++)
            {
                if(matrix[i][j]>=0)     cout<<matrix[i][j]<<"  ";
                else cout<<'m'<<"  ";
            }
            cout<<endl;
        }
        return 0;
    }
```

**【问题扩展】** 以上程序有一个问题,不论何时执行,总是产生一样的二维矩阵。其原因在于随机数产生函数 rand()。rand()函数返回 0 到 RAND_MAX 之间的伪随机数(RAND_MAX 常量被定义在 C/C++ 语言的头文件中)。rand()函数执行时会先执行一些初始化操作,直接使用 rand()函数则每次初始化都是一样的,因此产生的随机数序列也是一样的。为改变以上状况,需要使用 srand()函数。其原型为:

```
void srand (unsigned int n);
```

srand()函数使用自变量 n 作为种子,用来初始化随机数产生器。只要把不同的值传入 srand(),然后调用 rand()时,就会产生不同的随机数序列。因此,可以把系统时间作为 srand()函数的实际参数,这样就可以避免重复的发生。如果调用 rand()之前没有先调用 srand(),就和事先调用 srand(1)所产生的结果一样。比如以下程序段。

```
srand(1);                              /* 指定种子的值为 1 */
for (int i=0; i<10; i++)
    cout<<rand()%10<<" ";
```

每次运行都将输出:1 7 4 0 9 4 8 8 2 4

而以下程序段,由于把系统时间作为 srand()的参数,每次运行都将输出不同结果。

```
srand((unsigned)time(0));
for (int i=0; i<10; i++)
    cout<<rand()%10<<" ";
```

注意,为了使用上面程序段的 time 函数,需要在程序头部增加下列语句:

```
#include <ctime>
```

time(0)返回的结果是从 1970 年 1 月 1 日零时零分零秒到目前为止所经过的时间,单位为秒。

请根据这里讲解的方法修改本例题程序,改为随机数可以随时间的变化而变化。

## 2.11.5 表达式计算

编写程序计算一个由整数、加号、减号、小括号构成的算术表达式,假定表达式按字符

串读入，无多余空格且写法符合一般数学规律(既不考虑不合法的表达式)。下面是一个运行样例。

```
请输入表达式：1+((7-2)+(35-(4+1))+4)
1+((7-2)+(35-(4+1))+4)=40
```

**【本题目的】** 练习递归算法、循环控制、字符型数字转化为整型数字。

**【问题分析】** 首先考察不带括号的表达式，例如 7＋3－5＋6－1 等。回顾以前计算 1＋2＋3＋…＋100 的方法，可以想到不带括号的表达式的计算可用循环实现。只是循环中除了读数字，还要读符号(无符号则默认为正)。假定表达式存储在字符数组 p[]中，则可写出如下算法伪代码：

```
设计算结果 left=0,数组 p起始位置 loc=0;
循环(p[loc]不是结束标志'\0')            //表达式未结束
{
    //在下面过程中,位置 loc根据需要向后移动
    读取符号到变量 s中,s取 1或-1;      //无符号则取为 1
    读取后面的字符形式的数字,转换为整型放到 num;
    left=left+s*num;                   //数字累加到 left 中
}
```

再考虑有括号的表达式。假定要计算表式 $E_0=15-(7-(3+1))$，不妨设表达式 $E_1=7-(3+1)$、$E_2=3+1$。于是有 $E_0=15-(E_1)$、$E_1=7-(E_2)$。也就是说，为了计算 $E_0$，需要先计算 $E_1$；为了计算 $E_1$，需要先计算 $E_2$，而 $E_2$ 是标准的不带括号的表达式，可以用上面的算法计算。当 $E_2$ 计算完成后，将数字回带到 $E_1=7-(E_2)$中，只要去掉括号，这个表达式就是另一个标准的不带括号的表达式。这个过程是一个递归调用的过程，只要在上面的算法中稍加修改，当遇到'('时，递归调用表达式处理函数；当遇到')'时，返回子表达式计算结果给上一层函数即可。算法伪代码如下：

```
全局变量 loc指示 p的处理位置,初值为 0;
Calc(p)                                //计算 p[]中从 loc开始的子表达式
{
    设计算结果 left=0;
    循环(p[loc]不是')'且 p[loc]不是结束标志'\0')
    {
        //在下面过程中,位置 loc根据需要向后移动
        读取符号到变量 s中,s取 1或-1;//无符号则取为 1
        若(p[loc]是'(')                 //符号后面仍是表达式
        {
            loc++;
            用 Calc(p)计算子表达式,结果放到 num;
            left=left+s*num;
        }否则{
            读取后面的字符形式的数字,转换为整型放到 num;
```

```
            left=left+s*num;            //数字累加到 left 中
        }
    }
    子表达式计算完成,返回结果 left;
}
```

注意,在上面的算法中使用了全局变量 loc 指示表达式字符串 p 目前处理到哪个位置。这是因为表达式是层层嵌套放在一个字符串 p 中,当下层函数计算完成后要通知上层函数已经处理到哪个位置了,以便上层函数从此位置继续计算。为简单起见,这里将字符串目前处理到的位置 loc 作为全局变量。

**【程序参考】**

```
#include<iostream>
using namespace std;
int loc=0;
//计算从 p[loc]开始的子表达式,结果返回给上层函数
int calc(char *p)
{
    int left=0,num;
    while(p[loc]!=')'&&p[loc]!='\0')
    {
        int s=1;                        //s为当前数字或子表达式前的符号
        if(p[loc]=='+'){
            s=1;
            loc++;                      //跳过"+"
        }
        if(p[loc]=='-'){
            s=-1;
            loc++;                      //跳过"-"
        }
        if(p[loc]=='(')                 //遇到子表达式
        {
            loc++;                      //跳过"("
            num=calc(p);                //计算子表达式
            left=left+s*num;            //子表达式结果累加到 left 中
        }else{                          //符号后面是数字
            num=0;
            while(p[loc]>='0'&&p[loc]<='9')
            {   //字符型数字转化为整型数字
                num=num*10+p[loc]-'0';
                loc++;
            }
            left=left+s*num;            //数字累加到 left 中
        }
    }
    if(p[loc]==')')  loc++;  //跳过")",走到下一位置
```

```
    return left;                      //将结果传递给外部
}

int main()
{
    char e[200];
    cout<<"请输入表达式：";
    cin>>e;
    loc=0;
    cout<<e<<"="<<calc(e)<<endl;
    return 0;
}
```

**【问题扩展】** 即使不考虑计算乘法和除法的问题，上面的计算表达式的算法也有很多可改进的余地。

首先是可以增加对实数的处理。上面的程序已经可以处理整数部分的字符串，处理完整数部分并将其累加到结果变量中后，可以再增加下面的算法以便处理小数部分。

```
若(p当前位置是'.')                      //说明还有小数部分
    {
    跳过'.';
    num=0.;                            //累加器再次清零,num应为 double 型
    设 k=-1;                           //指数 k为 10的-1次方
    当(p当前位置字符在'0'和'9'之间)
    {
        p中当前字符转换为数字,再乘以 10^k 后累加到 num;
        走到 p的下一位置;
        k--;                           //指数 k减去 1
    }
    将小数部分 s* num累加到 left中;
}
```

使用上面的算法，要注意累加器 num 应为实数型变量，不然无法累加小数。另外，为了计算 10 的乘方，应包含头文件 cmath。

另一个可以增加的功能是判断左右括号是否匹配。最容易理解的方法是增加一个函数，在表达式计算开始前先判断括号是否匹配。函数设置某个变量为 0，然后从前至后扫描整个表达式字符串。若遇到'('就将变量加 1，若遇到')'就将变量减 1。处理完成后，最终若变量为 0 则是合理的，否则说明左右括号不匹配。

其实在上面的函数中，不妨再增加一点处理，将表达式字符串中的空格去掉。一个简单的方法是另设一个字符串，将表达式中非空格字符复制过去，最后再将无空格的串复制回原来的字符数组即可。

请根据上面的描述修改本例题程序，实现处理实数、去掉表达式中的空格、判断括号是否匹配的功能。

# 第3部分

# 习 题 解 析



## 3.1　习题1　程序设计与C++概述

1. 输入长、宽(均为整数),计算矩形的面积。
2. 输入长、宽、高(为实数),计算长方体的表面积和体积。
3. 输入半径(实数),计算圆的周长和面积。
4. 编写程序,输入5个实数,计算它们的平均数。

【编程提示】 要说明用a,b,c等符号表示实数,使用格式:

```
double a,b,c;
```

5. 编写程序,打印矩形

```
****************************
*                          *
*                          *
*                          *
*                          *
****************************
```

【编程提示】 使用输出语句cout输出若干行字符串即可。中间空白部分是空格。

6. 编写程序,打印如下图所示的卡片,其中姓名和电话号码从键盘输入。

```
*************************************
    Wang Feng
    Xi'an Jiaotong University
   Add.
    No.28 West Xianning Road
    Xi'an China,710049
    Tel.86-29-82668888
*************************************
```

【编程提示】 姓名和电话号码是用户输入的信息,使用下列格式说明表示姓名和电话号码的符号。

```
char name[40];
```

```
char tel[20];
```

其中 char 是关键词，name 和 tel 是自己起的表示姓名和电话号码的符号，40 和 20 说明名字不超过 40 个字符。电话号码不超过 20 个字符，是可以改的，比如改为 50 或 30 等。

输入使用：

```
cin.getline(name,40);
cin.getline(tel,20);
```

输出使用：

```
cout<<name;
cout<<tel;
```

7. 输入 n，计算

$$y = \left(1 + \frac{1}{n}\right)^n$$

的函数值。

**【编程提示】** 计算 x 的 n 次方，可以使用 pow 函数，使用格式为：

```
pow(x,n);
```

这个表达式的值就是 x 的 n 次方。不过，求 x 的平方、立方时，常用 x * x 和 x * x * x，这比 pow()的效率要高。

pow 是一个系统函数，包含在头文件 cmath 中，在文件开头添加下面的一行程序：

```
#include<cmath>
```

另外，编程时，1 应写为 1.0，n 为整数。当 $n \to \infty$ 时，$y \to e$。试着输入几个较大(如 100，200，500)的数看计算结果。

8. 输入大于 0 的数 x，计算函数

$$y = \sin x - \ln x + \sqrt{x} - 5$$

的函数值。

**【编程提示】** 计算 sinx 值的函数是 sin(x)，计算 lnx 的函数值的函数是 ln(x)，计算 x 的平方根使用 sqrt(x)。如下列语句分别显示 sin 3.1、log 3.1 和 3.1 的平方根：

```
double x=3.1;
cout<<sin(x);
cout<<log(x);
cout<<sqrt(x);
```

这些函数也是系统提供的数学函数，需要包含头文件＜cmath＞。其他数学函数见第 4 部分。

9. 输入 x，计算

$$y = \frac{x}{\sqrt{x^2 - 3x + 2}}$$

的函数值。

**【编程提示】** $x^2$ 通过 x * x 来计算，3x 写成表达式时不能省略乘法运算符"*"，3 和 2 表示实数时在程序中应写成 3.0 和 2.0。开方用数学函数 sqrt()。

10. 输入 x，a 计算

$$y = \log_a(x + \sqrt{x^2 + 1}), \quad (a > 0, a \neq 1)$$

的函数值。

**【编程提示】** C++ 中没有以任意数 a 为底的对数函数，但对数有换底公式：

$$\log_a b = \log_n b / \log_n a$$

其中"/"表示除。

## 3.2 习题 2 简单信息的表达与运算

1. 按表 2-1 定义不同类型的变量，计算并显示不同类型的变量占的字节数，并尝试：与赋不赋值有关吗？将变量改为不同类型的常量呢？如 3、3.0，1.0E-4、'e'、"continue"、true、false 等。

**【本题目的】**

(1) 理解不同类型的数据在内存中的存储形式和占用的空间是不同的；

(2) 掌握查看不同类型的数据在内存中所占字节数的方法；

(3) 记住基本类型的数据在内存中所占的字节数。

2. 温度转换。输入华氏温度，用下列公式将其转换为摄氏温度并输出。

$$C = \frac{5}{9}(F - 32)$$

**【编程提示】** 注意，5/9 在程序中应写为 5.0/9.0，否则，得到的结果会是 0，因为在 C++ 中整型数和整型数的运算结果还是整数(不是零数时会直接取整数部分)。

3. 编程试求函数

$$y = \frac{\sin x^2}{1 - \cos x}$$

当 $x \to 0$ 时的极限。

**【编程提示】** 三角函数的值是通过数学函数 sin(x)(正弦)、cos(x)(余弦)来计算的(函数使用见第 4 部分)。输入的数值逐步变小，观察 y 的值的变化。不要输入 0。

4. C++ 中的库函数 sin(x)、cos(x)等三角函数，自变量的单位为弧度。请编写程序，用户输入角度，计算其正弦、余弦、正切(tan)和余切的函数值并显示出来。如果用到 π，请将其定义为符号常量。

**【编程提示】** 弧度的 π 对应的角度为 180°，那么换算关系就是：

$$弧度 = \frac{角度}{180} \times \pi$$

C++ 中没有 π 这样一个符号表示圆周率，所以可以自己设一个符号并且让它表示 3.1415926 这个数。由于 π 的值是不会变的，可以将它说明为常量，使用下面的语句之一：

```
#define   PI 3.1415926
const double PI= 3.1415926;
```

5. 编程实现，用户从键盘输入 3 个整数，计算并打印这三个数的和、平均值及平均值的四舍五入整数值。

**【编程提示】** 将双精度变量赋值给整型变量或使用(int)强制类型转换，将 double 数转换为 int 数，得到整数结果(下取整)。为实现四舍五入，可以将一个实数加 0.5，再取整。如：1.6＋0.5，取整得到 2；1.2＋0.5 取整得到 1。

6. 找零钱。为顾客找零钱时，希望选用的纸币张数最少。例如 73 元，希望零钱的面值为 50 元 1 张，20 元 1 张，1 元 3 张。设零钱面值有 50 元、20 元、10 元、5 元和 1 元。请编写程序，用户输入 100 以下的数，计算找给顾客的各面值的纸币张数，并在程序中想一个验证结果是否正确的办法。

**【编程提示】** 找零钱。

73 除 50 得到的整数就是所找的 50 元纸币的张数，得到的余数就是找 50 元后剩下的零钱。

再用剩下的零钱除 20，整数就是二十元纸币的张数，得到的余数就是再找 20 元后剩下的零钱……如此继续。

求商时，直接将两个整型数相除，得到的就是商(整数部分)。

使用%运算符可得到两数相除的余数。

7. 小写转大写。用户输入小写字母，程序输出对应的大写字母。

**【编程提示】** 大写字母 A 的 ASCII 值为 65，小写字母 a 的 ASCII 值为 97，它们相差 32，所以小写字符转换为大写字母只要减 32 即可，而且 32 还可以用'a'－'A'得到。若 c 表示小写字符，则 c－'a'＋'A'就是相应的大写字母；若 c 是一个大写字母，则 c－'A'＋'a'就是相应的小写字符。表示字符的 c 用 char 说明，即 char c；。

8. 打印 ASCII 码。输入一个字符(可能为字母、数字或标点符号等)，在一行中打印该字符及该字符的 ASCII 的十进制、十六进制形式和八进制形式，数据之间用'\t'分隔。

**【编程提示】** 十进制、八进制和十六进制的控制字符为 dec、oct 和 hex，先输出一个控制符，以后输出的整数就以这样的数制的形式输出。例如，

```
cout<<oct<<v;
```

显示整型变量 v 的八进制形式。

9. 用户输入不超过 255 的 4 个数，将这 4 个数从左向右顺序保存在一个整型变量的 4 个字节中，输出这个整型变量值的十进制和十六进制形式。

**【编程提示】** 由于每个数不超过 255，所以它只用了整型变量 4 个字节中的最低一个字节。设保存 4 个数的变量为 a。先给 a 赋 0，将第 1 个数与 a 做按位“或”运算，则第 1 个数的低八位放到 a 的低八位；再将 a 左移 8 位，将第 1 个数移到了 a 的第 3 个字节(从左数)；然后与第 2 个数做按位“或”运算，第 2 个数放到的 a 的最后一个字节；再按相同的方法放第 3、第 4 个数；最后第 1、2、3、4 个数依次放在了 a 的第 1、2、3、4 个字节中。用十六进制形式输出，可以清楚地看到每个字节中的数。例如输入的 4 个数是 255、15、14、

13,则结果的十六进制形式为 ff0f0e0d。每两位是一个字节,这四个字节中的数就是输入的四个数的十六进制形式。

移位也可以用乘除运算实现。乘 2 相当于左移一位,除 2 相当于右移一位。乘 16 相当于左移 4 位。左移 8 位乘几呢?

10. 用户以字符形式输入 4 个数字字符,将其组成一个 4 位的整数。例如,用户输入:2 0 1 1,输出结果为 2011。注意,输入的 4 个数字是字符型,用 4 个字符型变量存储,而 2011 是由它们构造出的一个 4 位整数,用一个整型变量表示。

**【编程提示】** 一个整数 234,可以写为(((0＊10+2)＊10)+3)＊10+4。若 a=0;

```
a=a*10+2;  //得 a=2
a=a*10+3;  //得 a=23
a=a*10+4;  //得 a=234
```

这是常用的构造除一个整数的方法。第 3 章学习了循环以后,任何位数的整数都可以这样构造出来。

## 3.3 习题 3 运算的流程控制

1. 编程求三个数的最大数。

**【编程提示】** 求最大数一般需要逐个比较,谁大就保留谁。设用变量 max 表示当前的最大数。开始时 max 等于第一个数,然后将 max 与第二个数比较,如果第二个数比 max 还大,就将第二个数赋给 max,然后再将 max 与第三个数比较……

**【测试指南】** 输入 1 2 3;3 2 1;1 3 2;2 3 1;-2 -3 -1 等最大数在不同位置、不按顺序排列的多组数,检验执行结果。

2. 编程计算下列分段函数的值:

$$y=\begin{cases}x^2 & (x<0)\\ x^3+2x^2+x & (x\geqslant 0)\end{cases}$$

**【编程提示】** 本题主要练习分支语句的编写,注意大于等于使用的符号是“>=”,表达式中平方用两个 x 连乘,立方用三个 x 连乘,乘号不能缺省。

**【测试指南】** 输入整数、负数、0 检验运行结果。

3. 编程计算 1+2+3+…+n 的和,n 由用户输入。

**【编程提示】** 数学上,等差级数的和可以用公式计算。程序设计中求若干项的和一般是逐项相加的。设 sum 表示和,初始为 0,sum=sum+i,i 从 1 循环到 n 时,sum 就是所求的和。

**【算法描述】**

sum=0;
输入 n;
对 i=1,…,n
    sum=sum+i

输出 sum。

4. 编程计算 n!。n 由用户输入,输入的 n 不合法时给出提示。

**【编程提示】** 求 n! 和求 1 到 n 的和类似,只不过一个是加,一个是乘。由于 0 乘任何数等于 0,sum 的初始值不能为 0,应为 1。

**【算法描述】**

```
product=1;
输入 n;
对 i=1,…,n
    product=product * i
输出 product。
```

**【测试指南】** 输入负数、0、1、其他整数、实数,检验程序运行结果。

5. 编程计算前 N 个奇数的和并显示表达式,N 由用户输入。如 N=4,运行结果为 1+3+5+7=16。

**【编程提示】** 第 1 个奇数是 1,第 2 个奇数是 3……加 2 就是下一个奇数。用循环控制个数,每次循环加一个奇数,显示加号(最后一次不显示加号),最后显示等号和 N 个奇数的和。

**【算法描述】**

```
输入 N
sum=0;
k=1;
对 i=1,…,N
    sum=sum+k;
    如果 i!=N
        显示 k 的值和"+"号;
    否则
        仅显示 k;
    k=k+2;
显示"="号和 sum。
```

**【测试指南】** 输入负数、0、整数、实数,检验程序运行结果。

6. 编写程序,打印九九乘法表,形式如下:

```
1*1=1
1*2=2   2*2=4
1*3=3   2*3=6   3*3=9
1*4=4   2*4=8   3*4=12  4*4=16
1*5=5   2*5=10  3*5=15  4*5=20  5*5=25
1*6=6   2*6=12  3*6=18  4*6=24  5*6=30  6*6=36
1*7=7   2*7=14  3*7=21  4*7=28  5*7=35  6*7=42  7*7=49
1*8=8   2*8=16  3*8=24  4*8=32  5*8=40  6*8=48  7*8=56  8*8=64
1*9=9   2*9=18  3*9=27  4*9=36  5*9=45  6*9=54  7*9=63  8*9=72  9*9=81
```

注意,不能以字符串常量的形式显示,要用循环。

**【本题目的】** 练习循环的嵌套使用,循环变量的使用和控制。

**【编程提示】** 九九乘法表有 9 行,第 i 行有 i 列。用一个循环控制 9 行,i=1,…,9;在这个循环体内,再嵌套一个循环,称为内层循环,控制列,j=1,…,i,这就是 i 列。

每列显示列号 j、乘号、行号 i、等号和乘积,不换行。一行显示完再换行。为了使列对齐,每显示一列信息,输出一个控制符"\t"。

**【算法描述】**

N=9;

对 i=1,…,N

    对 j=1,…,i

        显示 j、“*”号、i、“=”号、j*i;

    换行

结束

7. 比较两个字符串的大小。用户输入两个不含空格的字符串,比较它们的大小,并显示出来,如 width>wide, wide<width,wide=wide。要求使用字符数组表示字符串,不能使用系统函数。

**【编程提示】** 字符串比较是按它们在词典中的顺序比较的,后面的为大。实际是逐个比较它们的 ASCII 值。如果第 1 个字母相等,则比较第 2 个字母;第 2 个字母还相等,再比较第 3 个……直到字符串结束,表示两个字符串是一样的(相等);或遇到不同的字母。

**【算法描述】** 两个字符串用字符数组 str1,str2 表示。

i=0

当 str1[i]==str2[i],且 str1[i]!=0,str2[i]!=0,时,循环

    i++;

K= str1[i]-str2[i]

如果 K>0

    表示 str1>str2

否则 如果 K<0

    表示 str1<str2

否则

    表示 str1=str2

结束。

注意其中的分支可以使用如下格式:

```
if(<条件>)
        {  <if块>   }
else if (<条件>)
    {  <else if块>}
else
    {  <else块>   }
```

**【测试指南】** 应输入相等、大于、小于、长短不一的两个字符串进行测试。

8. “张村有个张千万，隔壁九个穷光蛋，平均起来算一算，人人都是张百万。”平均数作为“一般水平”的特征有它的局限性。另一个反映“平均水平”的统计量是“中位数”。将序列的值按大小顺序排列起来，处于中间位置的数称为中位数。当项数 N 为奇数时，中间位置的数即为中位数；当 N 为偶数时，中位数则为处于中间位置的 2 个数的平均数。编程实现下列功能：用户从键盘输入若干数，计算它们的和、平均值、最大值、最小值和中位数。用户输入的数量不超过 100 个，但每次输入的个数不定，以－9999 作为结束标志（它不参与统计）。

**【问题分析】** 本题有一定的综合性。若干个数可以用数组表示，通过循环输入。初始元素的个数设为 0，每输入一个数，个数加 1，直到输入的数是－9999 表示输入结束。和、平均值、最大数、最小数分别用一个变量表示，逐个检查每一个数，求和，比较，最后计算平均值，找到最大、最小数。然后进行排序。N 为奇数，下标是 N/2 的数是中位数；N 为偶数，下标是 N/2 和 N/2－1 的数的平均值是中位数。

N 个元素的排序，冒泡排序。

**【算法描述】** 设用 sum、avg、max、min、mid 表示上述待计算的数，元素个数为 N，存放在数组 A 中。

```
(1) 输入数据
N=0
输入 A[N]
当 A[N]!=(-999)时,循环
    sum=sum+A[N];
    N++;
avg=sum/N
(2) 求最大、最小数
max=A[0],min=A[0]
对 i=1,…,N
    如果 A[i]>max
        max=A[i]
    如果 A[i]<min
        min=A[i]
(3) 排序,求中位数
对 i=1,…,N-1
    对 j=0,…,N-1-i
      如果 A[j]>A[j+1]
        交换 A[j]和 A[j+1]的值
结束
如果 N%2==0
  mid=(A[N/2]+A[N/2-1])/2
```

否则

mid＝A[N/2]

(4) 显示结果

如果 N＝0,

显示"没有输入任何元素"

否则

显示元素个数、和、平均数、最大数、最小数、中位数。

9. 求 π 的近似值。将 arctan(x)在 x＝0 展开,得

$$\arctan(x) = x - \frac{x^3}{3} + \frac{x^5}{5} - \frac{x^7}{7} + \cdots = \lim_{i=1}(-1)^{i+1}\frac{x^{(2i-1)}}{2i-1}$$

当 x＝1 时,arctan(x)＝π/4,从而这个级数即可以计算 arctan(x)的近似值,又可以计算 π 的近似值:

$$\pi = 4\left(1 - \frac{1}{3} + \frac{1}{5} - \frac{1}{7} + \cdots\right)$$

利用上式编程计算 π 的近似值,精确到小数点后 8 位。

**【问题分析】** 这是一个级数的求和。对于级数的求和问题,一般是设和的初始值为 0,构造一个通项。若不满足精度要求,则将其加到和中;然后再构造下一个通项,再检查精度。若通项 u(n)趋于 0,常以通项 u(n)＜ε 为终止条件,反过来 u(n)＞ε 就是循环条件。对本题,若 v 为通项的分母,则 v＋2 是下一项的分母。1/v 就是通项的绝对值。

每一通项的符号不同,是交叉的。数学上用$(-1)^{i+1}$表示。而若初始 sign＝1,则 sign ＝(－1) * sign 就可实现正负交替。

**【算法描述】** 设 pai 表示求出的近似值。

eps＝1.0e－8;

pai＝0

u＝1

sign＝1

当 1.0/u＞eps 循环

pai＝pai＋sign * 1.0/u

u＝u＋2;

sign＝sign * (－1)

pai＝pai * 4.0

输出 pai。

**【编程提示】** 编程时:(1)注意混合运算。若 u 为整数,1/u 要写为 1.0/u。(2)输出时,尽管设置的精度为 1.0e－8,但系统默认显示 6 位有效数字,为了能够显示到小数点后 8 位,在输出 pai 之前书写下列语句:

```
cout.setf(ios::fixed);
cout.precision(8);
```

10. 输入 n(n＜13)，计算 1!＋2!＋3!＋4!＋…＋n!

**【问题分析】** 本题也是级数的求和问题，通项是 n!。比较前后两个通项，u(n)＝n * u(n－1)，所以不需每项重新计算阶乘，只要在前一项上乘以 n 即可。

**【算法描述】** sum 表示级数的和，u 表示通项。

```
输入 n
sum＝0
u＝1
对 i＝1,…,n
    u＝u * i
    sum＝sum＋u
输出 sum
```

**【测试指南】** 输入正数、负数、0、实数进行测试。输入的数最大能达到多大？为什么？

**【问题扩展】** 修改程序，使其能显示级数的表达式，如输入 5，显示：

```
1!+ 2!+ 3!+ 4!+ 5!= 153
```

11. 对给定的正整数，不计算 n! 的值，统计 n! 中末尾 0 的个数。

**【问题分析】** 本题不需要计算出 n! 是多少，只需要说 n! 的值末尾有几个 0。有一个 0 就有一个因子 10，有两个 0 就有两个因子 10(10 * 10)。10 的因子为 2 和 5，就是说，如果 n! 中出现一对因子 2 和 5，就会产生一个 0。可以想像 n! 中因子 2 出现的次数比 5 出现的次数多，所以，只要统计出 n! 中有多少个因子 5，也就是有多少个 0。

**【算法描述】**

```
count＝0;
对 i＝5,…,n,间隔 5
    a＝i
    while(a!＝0 && a%5＝＝0)      //a 能被 5 整除
          count＋＋                //因子个数加 1
           a＝a/5                  //除掉一个因子
输出 count
```

12. 1202 年，意大利数学家列奥纳多(Leonardo Pisano，外号 Fabonacci 斐波那契)，在他的《算盘全书》中描述了一个关于兔子繁殖的问题。如果一对兔子每月能繁殖一对小兔，每对小兔在第三个月成熟，又能生出一对小兔。假定在不死亡的情况下，由一对小兔开始，10 个月后有多少对兔子？

用下表分析，不难发现每月兔子的数量依次为：1，1，2，3，5，8，13，21，…，每月的兔子数量是前两个月的和，这就是 Fabonacci 序列。写成公式为：

$$F_0=0,\quad F_1=1,\quad F_n=F_{n-1}+F_{n-2}$$

| 月份 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 |
|---|---|---|---|---|---|---|---|---|---|---|
| 成熟的兔子 | | 1 | 1 | 2 | 3 | 5 | 8 | 13 | 21 | 34 |
| 出生的兔子 | 1 | 0 | 1 | 1 | 2 | 3 | 5 | 8 | 13 | 21 |
| 总数 | 1 | 1 | 2 | 3 | 5 | 8 | 13 | 21 | 34 | 55 |

(1) 编程计算 Fibonacci 序列的第 n 项和前 n 项的和,n≥0,由用户输入。

**【问题分析(1)】**

Fibonacci 序列的递推公式是 $F_n = F_{n-1} + F_{n-2}$。

若 f0 是前一项,f1 是当前项,则 f2=f0+f1 就是后一项。

再将 f2 看作当前项,f1 看作前一项,则 f1+f2 就是后一项。这一计算方法的公式可以写为:

```
f0=f1          //将 f1 看作前一项
f1=f2          //f2 看作当前项
f2=f0+f1       //f1+f2(是现在的 f0、f1 了)就是后一项
```

这种“将**项看作**项”的做法是程序设计中常用的手法,这样就不需声明很多变量,也不需使用数组,可以很简洁、很快地求解问题。

至于求和,还是逐项累加。

**【算法描述(1)】**

```
输入整数 N
fn=0
sum=0
如果 N<0,
    显示“输入错误,N 应为大于等于 0 的整数”;
否则  如果 N=1,
      则 fn=1
      sum=1
否则  如果 N>1
      f0=0
      f1=1
      sum=1
      对 i=2,…,N
          fn=f1+f0
          sum=sum+fn
          f0=f1
          f1=fn
输出第 N 项 fn 和前 N 项的和 sum。
```

**【测试指南(1)】** 输入正数、负数、0、1、2 等数据,检验结果的正确性。

(2) $F_{n-1}/F_n$，n=1,2,3…，又构成一个序列 0,1/1,1/2,2/3,3/5,5/8,…。当 n 较大时，$F_{n-1}/F_n$ 接近于哪个数？可以以相邻两数的差的绝对值小于 10E−8 为结束条件，打印 n、第 n 项的值及前 n 项的和。

**【问题分析(2)】** 这个序列常称为分数序列。

若 $F_{n-1}$ 和 $F_n$ 是 Fibonacci 序列的相邻两项(第 n−1 和第 n 项)，则 $F_{n-1}/F_n$ 就是分数序列的第 n 项，所以这个序列的通项是可以用 $u_n=F_{n-1}/F_n$ 来计算的，n=1,2,3…。

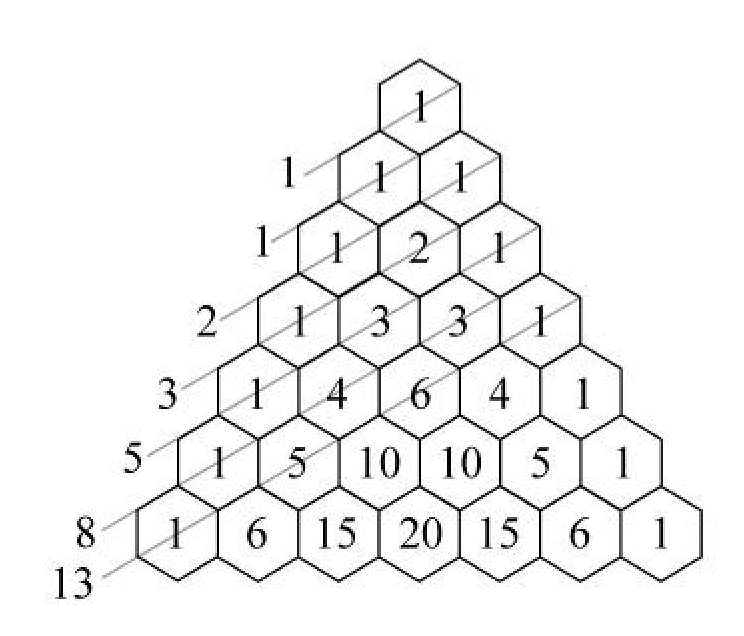
$u_{n-1}=F_{n-2}/F_{n-1}$，$u_n=F_{n-1}/(F_{n-1}+F_{n-2})$。设 $a=F_{n-2}$，$b=F_{n-1}$，则 $u_{n-1}=a/b$，$u_n=b/(a+b)$，这就是前后两项的关系，即知道了前一项的分子和分母，就可以计算出后一项的分子和分母。

对 $u_n=b/(a+b)$ 来说，b 就是新的 a，(a+b)就是新的 b，再计算后一项。

**【算法描述(2)】** eps=1.0E−8

```
a=0                        //第1项的分子
b=1                        //第1项的分母
u0=-1                      //第0项，设为-1可避免提前终止
u1=0                       //第1项
sum=0                      //和
n=1                        //项号
当|u0-u1|>eps 时循环
    tmp=a
    a=b                    //新一项的分子
    b=(tmp+b)              //新一项的分母
    u0=u1                  //新项变旧项
    u1=a/b                 //新一项的值
    n=n+1
    sum=sum+u1             //求和
输出 n、u1 和 sum。
```

(3) 下图六边形中的数字组成杨辉三角形，每一行是$(a-b)^n$，展开式的系数，n=0,1,2,…。沿图斜向的灰线将画到的数字加起来，这个序列又是什么呢？请编程打印杨辉三角形。




**【问题分析】** 本题打印出左对齐的杨辉三角形即可。

```
1
1  1
1  2  1
1  3  3  1
……
```

本题的第 n+1 行是$(a-b)^n$展开式的系数。通过直接计算系数的方法求各行的数字是可以的,但一般需要计算组合数,而计算组合数需要计算阶乘。在 C++ 中,用整型变量保存阶乘,只能保存最大 12 的阶乘。大家知道杨辉三角形的规律是一行中第一个和最后一个数是 1,其他是上方两数的和。利用这一规律,可以简化计算。方法是用两个数组,一个表示上一行的各数,相邻两个数相加就是下一行的各数。

**【算法描述】** 数组 A1 保存上一行的系数,数组 A2 保存当前行的系数。

```
A1[0]=1
N=1
输入整数 N>0                //N 表示行数
如果 N=1
    输出 A1[0],换行
否则
    输出 A1[0],换行
    对 k=2,…,N
        A2[0]=1
        A2[k-1]=1
        对 i=1,…,k-2
            A2[i]=A1[i-1]+A1[i]
        输出 A2[i],i=0,…,k-1
        A1[i]=A2[i],i=0,…,k-1
结束
```

**【思维扩展】** 本题揭示了一些问题的内在规律。兔子的繁殖问题是一个生物繁衍进化的问题,表现出规律性,这就是 Fibonacci 序列。而相邻两个数的比值随着 n 的增加,趋于黄金分割数(比)0.618……第(3)题中,按照斜线将杨辉三角值的数相加,这也是 Fibonacci 序列。自然界就是这样的神奇。

13. 求 a+aa+aaa+aaaa+…+aa…a(第 n 项,n 个 a),其中 a 是 1～9 的整数。例如,a=1,n=3 时,式子为 1+11+111;当 a=5,n=6 时,式子为 5+55+555+5555+55555+555555=617280。

**【问题分析】** 设级数的通项为 u(n),u(1)=a,u(n)=u(n-1)*10+a,由此容易计算和。

**【算法描述】**

```
输入 a
输入项数 N(>0)
```

```
u=a
sum=u
对 i=2,…,N
    u=u*10+a
    sum=sum+u
输出 u
```

**【测试指南】** 输入 a=1,n=1,2,3,4,5,6;a=2,n=2,3,4,5,6;

**【问题扩展】** 修改程序,使其显示算式,如输入 a=5,N=6,显示结果为:

```
5+ 55+ 555+ 5555+ 55555+ 555555= 617280
```

14. arcsin(x)写成级数形式为:

$$\arcsin(x) = x + \frac{x^3}{2 \cdot 3} + \frac{1 \cdot 3 \cdot x^5}{2 \cdot 4 \cdot 5} + \cdots + \frac{(2n)!\,x^{2n+1}}{2^{2n}(n!)^2(2n+1)} + \cdots$$

用户输入 x,利用该式,计算反正弦函数的值。结束条件可以设为$|u|<\varepsilon$,其中 u 为通项。

**【问题分析】** 这也是一个级数的求和问题。通项的计算可以去计算乘方和阶乘,但阶乘的计算即使 n 很小,值也会很大,大到整型变量甚至实型变量无法准确表示,使误差很大。级数求和的最好方法是利用前一项计算后一项。设:

$$u_n = \frac{(2n)!\,x^{2n+1}}{2^{2n}\,(n!)^2(2n+1)}$$

则

$$u_{n-1} = \frac{(2n-2)!\,x^{2n-1}}{2^{2n-2}\,((n-1)!)^2(2n-1)}$$

两项相除 $u_n/u_{n-1}$,经过推导可以得到:

$$u_n = \frac{(2n-1)^2 x^2}{2n(2n+1)} u_{n-1}$$

这样就避免了阶乘和高次方的计算。

**【算法描述】** eps=1.0e-8

```
输入 x
n=0
u=x
asinx=u
当|u|<eps 时循环
    n=n+1
    u=u*(2n-1)*(2n-1)*x*x/(2n*(2n+1))
    asinx=asinx+u
输出 asinx
结束。
```

**【编程提示】** (1)注意 2n 在程序中应写为 2*n。(2)注意整数的除法运算应先进行类型转换。

15. 猴子吃桃问题。第一天,猴子摘下一堆桃子,当天吃了一半,感觉没吃够,又饶了一个。以后每天如此,到第10天的时候,发现只剩下一个桃子了。编程计算第一天猴子摘了多少桃子。

**【问题分析】** 设a(n−1)是前一天未吃之前的桃子数,则a(n)=a(n−1)/2−1,反过来a(n−1)=(a(n)+1)*2,a(10)=1,n=9,8,7,…。当n=1时就是第1天摘得桃子数。

**【算法描述】**

n=10
a=1;
对n=9,8,…,1,间隔−1
    a=(a+1)*2
输出a

16. 谁是小偷。某小区发生盗窃案,有4个人嫌疑最大,警察找他们来讯问,

A说:不是我。

B说:是C。

C说:是D。

D说:他冤枉人。

三人中有一人说了假话,请编程分析谁是小偷。

**【问题分析】** 这是一道推理题。人工解该题时的一般做法是假设A是小偷,看是否符合"三人说真话"的条件;再假设B是小偷,看是否符合"三人说真话"的条件……总之是假设某人是小偷,再对4人说的话进行判断,看哪种假设符合"三人说真话"的基本前提。那么问题就是如何表示这4个人,如何表示小偷,如何表示每人说的话,如何判别真假。

用1,2,3,4表示4个人,thf表示小偷。若thf=1,表示A是小偷。4人说的话用下式表示:

```
thf!=1
thf==3
thf==4
thf!=4
```

说真话的表达式的值为1(true),说假话的表达式的值为0(false)。thf=1,2,3,4就是4种假设,看哪种假设下,上述4个条件表达式的和为3。

**【算法描述】** 1,2,3,4表示四个人,thf表示小偷。

对thf=1,2,3,4
    如果(thf!=1)+(thf==3)+(thf==4)+(thf!=4)==3
        输出char('A'+thf−1 )
结束

**【编程提示】** 算法中char('A'+thf−1)也是一个小技巧。如果直接输出thf,是数字,是某个人的数字代号,而char('A'+thf−1)将数字的1、2、3、4转换成了A、B、C、D。

**【测试指南】** 开始编推理题，为了验证是否正确，应先人工推理，再运行程序，看结果是否一致。不一致时，要分析到底是谁错了。不能迷信程序，也不能迷信自己。

**【思维扩展】**

（1）本题的解法不唯一，可以有各种各样的解法，主要是能判断对错。

（2）类似的题目很多，如：小刚、小华和小明三位同学中，有一位同学做了好事。老师问他们三人是谁做了好事？

小明说："是小华做的"。

小华说："不是我做的"。

小刚说："不是我做的"。

已知三人中有两人说的是假话．只有一人说的是真话。老师想了想，说："小刚做了好事不留名，是个好孩子"。

编程让计算机推理一下，不是很难了吧？

再找一些推理题，编程推理。

17．"香莲碧水动风凉，水动风凉夏日长。长日夏凉风动水，凉风动水碧莲香。"是清朝女诗人吴绛雪的回文诗，正读和倒读是相同的。写程序，判断一首诗是不是这样的回文诗（诗以一个字符串的形式输入，中间无标点）。

**【问题分析】** 设字符串用字符数组 str 表示。如果是字符串的回文，如"abcba"，则只要判断 str[i]与 str[N－1－i]，i＝0，…，N/2－1 是否相等即可（N 为字符串长度）。每个汉字占两个字节，要比较第一个汉字与最后一个字是否相等，需要比较 str[0]与 str[N－2]和 str[1]与 str[n－1]是否相等。

**【算法描述】**

```
输入汉字字符串 str(不能有字母和标点符号)
求字符串的长度 N
cycle=true
对 i=0,2,…,i<N/2-1,间隔 2
    如果 str[i]!=str[N-2-i] 或 str[i+1]!=str[N-1-i]
        cycle=false
        退出循环
输出 cycle                //1 表示是,0 表示不是。
```

**【问题扩展】** 修改程序使能判断输入的字符串中是否包含英文字符，如果有，则重新输入。对英文字符的判别，就是看码值是否在[0，127]之间。

18．整数里也有回文如 12321，正反顺序的数字都是 1，2，3，2，1，称为回文数。用户输入一个整数，判断是否回文数？

**【问题分析】** 一个整数如 123，倒过来写 321，它们不相等，这不是回文数；而 121 倒过来还是 121，就是回文数。这是整数回文数的判别方法之一。

**【算法描述】** 判别回文数。

```
输入正整数 N
a=N
```

```
RN＝0
当 a!＝0 时,循环
    g＝a%10              //取个位
    RN＝RN＊10＋g        //构造倒过来的数
    a＝a/10              //去掉个位
如果 RN＝N
    则该数就是回文数
    否则不是回文数
结束
```

编程找出十万以内的平方回数。回文数同时又是一个数的平方,如 676 是回文数,又是 26 的平方,称为平方回数。

**【问题分析】** 一个回文数,将其开方后取整,再平方,如果与原数相同,则是平方回文数。还有一种方法是计算一个数的平方,再判断平方数是否回文数。

请自己写出算法再编程。

**【问题扩展】** 一个回文数,如果还是一个数的立方,就是立方回数。编程探索有立方回文数吗？有 4 次方回数吗？5 次、6 次呢？

19. 任意一个大于 1 的正整数可以表达为一系列素数的乘积,这样的分解是唯一的,称为素数分解。例如,60 可以分解为 2×2×3×5。编写程序,显示用户输入的一个正整数的素数分解,输出格式形如:60＝2×2×3×5。

探究:RSA 公钥加密算法的基础就是大数的素数分解是困难的。你编写的程序能分解多大的素数呢？

**【问题分析】** 素数分解实际就是列出一个数的所有最小因子。可以从 2 开始验证这个数是否能被 2,3,4,…,整除。如果能,就从中去掉这个因子,直到这个数变为 1。

**【算法描述】**

```
输入 N
a＝N
k＝2
当 a!＝1 时循环
        当 a%k＝0 时循环
            输出 k
            a＝a/k
        k＝k＋1
结束
```

**【编程提示】** 上述算法只是说明如何分解,至于输出格式,请自己思考。

**【思维扩展】** 素数是一类很奇特也很有用的数。例如,有人研究说在汽车变速箱齿轮的设计上,相邻的两个大小齿轮的齿数最好设计成质数,可增强耐用度减少故障;在害虫的生物生长周期,质数次数地使用杀虫剂是最合理的,因为都是使用在害虫繁殖的高潮期,而且害虫很难产生抗药性;以质数形式无规律变化的导弹和鱼雷可以使敌人不易拦

截等。

RSA 公钥密码基于的理论是大数分解的困难性,例如:

1230186684530117755130494958384962720772853569595334792197322452151726400507263657518745202199786469389956474942774063845925192557326303453731548268507917026122142913461670429214311602221240479274737794080665351419597459856902143413

这是一个 232 位的十进制数,是两个素数的乘积。怎么表示,怎么分解呢?

# 3.4　习题 4　复杂信息的表达与处理

1. 编写程序,将 N(N<10)阶方阵转置,例如:

$$\begin{bmatrix}5&6&7&9\\2&8&5&4\\3&7&16&15\\1&4&8&11\end{bmatrix} \qquad \begin{bmatrix}5&2&3&1\\6&8&7&4\\7&5&16&8\\9&4&15&11\end{bmatrix}$$

转置前的方阵 A　　　转置后的方阵 A

**【问题分析】** 阶数小于 10,可以声明 10 * 10 的二维数组。转置是将下三角的元素与上三角的元素互换。设 i 是行标,j 是列标,则下三角的下标范围是 i=2,…,N,j=1,…,i-1。

**【算法描述】** 设矩阵用 A 表示。

输入阶数 N

输入 N 行 N 列的元素

对 i=1,…,N,间隔 1

　　对 j=1,…,i-1,间隔 1

　　　　交换 A[i][j]和 A[j][i]

按行输出 A 的元素

结束

**【编程提示】** 程序应使原来的矩阵 A 转置,而不是再声明一个 B 是 A 的转置,更不是按列输出 A 的元素,因为那样 A 仍是没有转置的,实现方法也不一样。

**【测试指南】**

(1) 输出结果后,仔细观察得到的是否是已转置的矩阵。

(2) 输入不同阶数的矩阵验证。

**【问题扩展】** 修改程序,使其能对长方阵转置。

2. 设有一个有序的整型数组,数据元素从小到大排列,初始时数组中没有元素。用户从键盘输入一个整数,将其插入到数组的合适位置,使数组保持有序,按顺序显示插入后的数组元素和插入的位置。程序要考虑对数组满的情况的处理。

**【问题分析】** 如果数组是空的,输入一个元素,就将其放入 A[0]即可。若数组中有 A[0],…,A[n-1]等 n 个元素,且是有序的,再输入一个元素 x,就要找一个合适的位置

插入该元素。插入的方法是先与 A[n－1]比较，如果 x＜A[n－1]，就将 A[n－1]后移一个位置，再继续与前一个元素比较。如果 x＜A[n－2]，再将 A[n－2]后移，直到 x 不小于前面的元素，就将 x 放在该元素的后面。

**【算法描述】** 设有数组 A[MAX]，元素下标从 0 到 MAX－1，N 表示当前元素的实际个数。

```
N＝0
循环
        输入 x
        如果 N＝MAX，跳出循环
        否则
                对 j＝N－1，…，0，间隔－1
                      如果 x＜A[j]，A[j＋1]＝A[j]
                      否则，结束本层循环
                A[j＋1]＝x
                N＋＋                    //实际元素个数加 1
                显示 A[j]，j＝0，…，N－1，插入位置下标为 j＋1
显示"数组已满"
结束
```

3. 打印杨辉三角形。

**提示**：使用二维数组，下一行的系数等于上一行的两个系数之和。

4. 将例 4-6 改为一维数组，实现相同的功能。

**【问题分析】** 例 4-6 实现的是矩阵的相乘运算，计算公式是：

$$c_{ij} = \sum_{k=1}^{N} a_{ik} \times b_{kj}, \quad i = 1,\cdots,M, j = 1,\cdots,K$$

使用一维数组存放矩阵时，矩阵 i 行 j 列的元素（从 0 开始）在一维数组中的存放位置是 i＊N＋j，其中 N 是矩阵的列数。设 A 存放 M×N 的矩阵，B 存放 N×K 的矩阵，C 是它们的乘积，M 行 K 列。这时的公式可以写为：

$$C[i*K+j] = \sum_{k=1}^{N} A[i*N+k] \times B[k*K+j], \quad i = 1,\cdots,M, j = 1,\cdots,K$$

**【编程提示】** 将原来程序中的 A[M][N]换为 A[M＊N]，A[i][j]换为 A[i＊N＋j]即可。输入、输出也类似。

5. 矩阵用一维数组存储，判断矩阵是否对称矩阵。

**【问题分析】** 判别矩阵的对称性，就是看下三角的元素是否与上三角的元素相等。设 A 为待判别的方阵，即比较 A[i][j]是否等于 A[j][i]，i＝1，…，N－1，j＝0，…，i－1（下标从 0 开始）。

**【算法描述】** 设一维数组存放矩阵元素 A。

```
输入维数 N
输入 N 行 N 列的矩阵元素
```

```
symmetry=true
对 i=1,…,N-1
    对 j=0,…,i-1              //对下三角的每一个元素(不含对象)
        如果 A[i * N+j]不等于 A[j * N+i]
                                //检查是否与上三角的对称位置的元素相等
            symmetry=false
            退出循环
    如果 symmtry=false
        退出循环
如果 symmetry=false
    显示"不是对称矩阵"
否则
    显示"是对称矩阵"
结束。
```

6. 编写程序,用户输入一个英文字符串,将其中的字符顺序反转过来(仍保存在原来的字符数组中),然后输出。例如,输入"student",输出"tneduts"。要求字符串用字符数组存放,不使用库函数。

**【问题分析】** 反转字符串就是将第 1 个字符和最后一个字符交换位置,第 2 个字符和倒数第 2 个字符交换位置……直到中间两个字符交换了位置。

**【算法描述】** 一维字符数组 str 存放待处理的字符串。

```
输入字符串存放到 str 中
计算 str 的长度 N               //自己编程计算,不允许使用库函数
对 i=0,…,[N/2]-1               //[]表示取整
    交换 str[i]和 str[N-1-i]
输出 str。
```

**【编程提示】** 本题反转后,字符串仍要存放在原字符数组中。如果先将字符串按倒过来的顺序放在另一个字符数组中,然后再逐个按顺序移回 str 中,也可以实现,但工作量要大一些。只逆序显示,原字符串本身没有反转的做法是不符合题目要求的。

**【测试指南】** 输入奇数个字符,输入偶数个字符进行测试。

7. 编写程序,去掉字符串末尾的空格符。要求字符串用字符数组存放,不使用库函数,结果要显示末尾有空格的字符串和没有空格的字符串。

**【问题分析】** 末尾有空格的字符串如"abc　　",没有空格的字符串如"abc"。字符串的结束标志是'\0'。如果在字符'c'之后加'\0',就是去掉了末尾的空格。然而问题是如何找到最后的、不是空格的字符呢?可以从后往前逐个字符串检查,直到遇到不是空格的字符,在其后放置结束符'\0'。

是否空格,可以直接比较,如 c 是字符变量,可以写 c==' '或 c==32。如果 c 是空格,表达式的值为 true,否则为 false。

空格在屏幕上显示的是空白,是不可见的。为了让用户看到确实有空格存在,可以在

字符串的前后，紧跟字符串显示两个其他字符，如竖线'|'等。

**【算法描述】** 设字符串用字符数组 str 表示。

```
输入字符串到 str 中
输出'|',str,'|'
计算字符串的长度 N
i=N-1
当 str[i]等于空格时，循环
  i=i-1
str[i+1]='\0'
输出'|',str,'|'
结束
```

**【编程提示】** 输入带空格的字符串，使用 cin.getline(str,100);。遇到空格不显示的做法不符合题目要求，因为内存中，字符串中的空格仍然存在。

输出字符串时前后加"|"很重要，否则，看不到空格，既不知原来输入的字符串是否真的有空格，也不知道是否真的去掉了空格。

**【测试指南】** 输入末尾带空格、不带空格、带 1 个空格、带两个空格和带多个空格的字符串进行测试。输入前、后和中间有一个、多个空格的字符串进行测试。

**【思维扩展】**

(1) 如果从字符串的开头检查，遇到空格时设置为'\0'，结果会是什么呢？

(2) 去掉、添加字符串末尾、开头的空格，甚至是中间的空格是文本信息处理(如信息检索和信息表示等)的基本操作。去掉它们常常是因为它们不包含本质的信息，添加它们常常是因为要区分信息的不同部分。

8. 编写程序，去掉字符串开头的空格符。要求字符串用字符数组存放，不使用库函数。

**【问题分析】** 开头有空格的字符串形如"          abc"，开头没有空格的字符串如"abc"，但末尾和中间仍可能有空格。去掉字符串开头的空格，将第 1 个不是空格的字符及其后所有的字符(包括结束符)前移。

**【算法描述】** 设字符串用字符串数组 str 表示。

```
输入前、后、中间带空格的字符串 str
输出'|',str,'|'
k=0;
当 str[k]等于空格时，循环
    k++
i=0;
当 str[i+k]不等于结束符时循环
    str[i]=str[i+k]
    i++
str[i]='\0'
```

输出'|',str,'|'

结束

**【编程提示】** str[i]='\0'很重要,去掉它试试看。

**【测试指南】** 要输入开头没有空格、有空格、有一个和多个空格的字符串测试,还要测试末尾有、无空格的字符串。

9. 编写程序,去掉字符中间的所有空格。要求字符串用字符数组存放,不使用库函数。

**【问题分析】** 中间有空格的字符串形如"abc　def　gh",中间没有空格的字符串形如"abcdefgh",但前后可能有空格。去掉字符串中间的所有空格,跳过开头的空格,找到中间空格部分的开始和结束位置,有几个空格,将后面的所有字符前移;然后还要查找是否中间还有空格,直到确定中间没有空格了。跳过中间的空格,只要从 str[0]开始检查,连续空格不作记录,遇到非空格符后,如果再遇到空格,且连续空格的末尾不是结束符,就是中间的空格。

**【算法描述】** 字符串用字符数组 str 表示。

```
输入字符串 str
输出'|',str,'|'
i=0
当 str[i]等于空格时循环                //跳过开头的空格
  i++
当 str[i]!='\0'时循环                 //检查中间的所有空格
    当 str[i]不等于空格时循环          //跳过不是空格的字符
        i++
    k1=i                              //中间空格的起始位置
    j=i
    当 str[j]等于空格时循环            //结束时 j 是空格的结束位置(非空格)
        j++
    如果 str[j]!='\0'                 //不是末尾的空格
        k2=j                          //结束位置
        j=0
        当 str[k2+j]!='\0'时循环      //前移后面的所有政府
            str[k1+j]=str[k2+j]
            j++
        str[k1+j]='\0'                //末尾加结束符
    否则
        跳出循环                      //是末尾的空格,不去,结束
    i++
输出'|',str,'|'
结束
```

**【问题扩展】** 上述算法是从前向后查找中间空格的位置,还可以从后向前查找中间

空格的位置。

10. 编写程序，在字符串中查找子字符串，找到则返回第一个字符所在的位置(从 1 开始)，找不到则显示"没有该子串"。要求字符串用字符数组存放，不使用库函数。

**【问题分析】** 设文本字符串为 str1，待查找的字符串为 str2，在 str1 中查找 str2。设 k 和 j 分别为两个字符串中的字符的序号，开始时 k=0，j=0，即从第 1 个字符开始比较。若 str1[k]!=str2[j]，则 k++，j==0，即从 str1 的下一个字符看是不是 str2；若 str1[k]==str2[j]，则 i=k，i++，j++，再比较 str1[i]和 str2[j]，即比较下一个字符……若 str2 比较到了末尾，说明找到了 str2，k 是起始位置；若比较到某个字符串时 str1[i]!=str2[j]，说明只是找到了 str2 的前几个字符串，即从 k 开始的字符串不是 str2，这时 k++，j=0，再继续比较 str1[k]和 str2[j]……

**【算法描述】** 文本字符串为 str1，待查找的字符串为 str2。

```
输入 str1，str2(均非空)
k=0
j=0
当 str1[k]!=0 且 str2[j]!=0 时循环
    如果 str1[k]!=str2[j]
        k++
    否则
        i=k
        当 str[i]==str2[j]且 str1[i]!=0，str2[j]!=0 时循环
            i++
            j++
        如果 str2[j]==0，找到，k 是 str1 中 str2 的起始下标
        否则，没有找到，k++，j=0
若 str1[k]=0，str2[j]!=0，则说明没有找到。
```

**【测试指南】** 测试的数据应有 str2 在 str1 开头、末尾、中间的，有找不到的，有仅和 str2 的开头几个相同的。例如可以输入以下字符串对进行测试。

```
asdf   asd
asdfg dfg
asasasdf asd
asdfg   fgh
asdfg www
```

11. 编写程序，将数字组成的字符串转换为整数，例如将"1757"(字符串)转换为 1757(整型数)，要能处理负数。

**【问题分析】** "1757"是字符串，1757 是整数。一个数字形式的字符转换为整数的方法是：

```
a=c-'0'         //c表示一个数字字符,如'0', '1', '2', '3'等
```

构造 1757 使用秦九韶算法,即(((0 * 10+1) * 10+7) * 10+5) * 10+7。

**【算法描述】** 数字字符串在 str 中。

```
输入 str
i=0
a=0
当 str[i]!=0 时循环
    a=a * 10+str[i]-'0'
输出 a
```

**【测试指南】** 输入不同位数的数字字符串,输入非数字字符串进行测试。

**【问题扩展】** 当用户输入的字符串中有非数字的符号时,计算可能出错。修改程序,在转换为整数前对字符串进行检验,如果是数字组成的,才进行转换,否则输出提示信息"字符串中有非数字字符"。

12. 编写程序,将字符串形式的数转换为实数(可能为正、为负、为实数、为整数)。如将"-16.24"(字符串类型)转换为-16.24(双精度类型)。

**【问题分析】** 对一个字符串形式的数,看第 1 位是否符号。若是"-"号,则说明是负数;若是"+"号或没有符号,则说明是正数。然后转换整数部分(与上题类似)。遇到小数点后,转换小数部分。小数部分的位权依次为 1/10,1/100,1/1000 等。位权逐项除 10。

**【算法描述】** 用 str 表示待转换的字符串,sign 表示符号,a 表示转换后的数。

```
输入字符串 str
sign=1
i=0
如果 str[i]是负号,sign=-1,i++
否则,如果 str[i]是正号,sign=1,i++
a=0
当 str[i]!='.'且 str[i]!=0 时循环
    a=a * 10+str[i]-'0'
    i++
如果 str[i]=='.',i++
b=10
当 str[i]!=0 时循环
    a=a+(str[i]-'0')/b
    b=b * 10
i++
输出 a
```

**【编程提示】** 作除法时,注意转换为实型再相除。

**【测试指南】** 输入有正号的数、有负号的数、无符号的数、无整数的数、无小数的数、以小数点开头和结尾的数、多个正号、符号、小数点及有其他非法字符的数验证程序。

例如下列数据:+123,-123,123,12.3,-12.34,0.12,12.0,.12,12.,--12,

＋＋12,1..10, 12＋2,12－2,12abc34 等。

**【问题扩展】** 算法依然没有描述合法性的检验,请自己添加对输入的字符串进行合法性检验的程序,保证转换的顺利进行和正确性。

13. 输入若干事件的名称和一天内的发生时间(时、分、秒,24 小时制),按顺序计算相邻两事件发生的间隔。时间的输入格式如 18 28 15,表示 18 点 28 分 15 秒,输入时无顺序。要求时间用结构体表示,显示按时间排序的事件,同时显示相邻事件的时间间隔,格式如下(下划线部分为输入数据):

本程序进行事件排序并计算事件间隔:

请输入事件名称和事件时间,格式为 事件名 时　分 秒:

```
class1 8 0 0
class3 14 30 10
class2 10 10 10
class4 16 40 10
0 0 0 0

class1   8:0:0
class2   10:10:10
         间隔:2时 10分 10秒
class3   14:30:10
         间隔:4时 20分 0秒
class4   16:40:10
         间隔:2时 10分 0秒
Press any key to continue
```

**【问题分析】** 关于时间的排序,可以先比较时,再比较分,最后比较秒。而对于时间的间隔,通常化为基准时间(如 0 时 0 分 0 秒)以来的秒数,相减,再转换为时、分、秒。由于转换为秒后的计算比较统一,避免了很多 if 语句和条件,所以排序也可以转换为秒数再排序。

时间用结构体描述,成员可以有时、分、秒和秒数。

事件可以用结构体描述,成员有事件名称、发生时间和间隔,其中发生时间是时间型的变量。

**【算法描述】** 用结构体数组 event 表示若干事件,N 表示事件个数。

N＝0

输入事件 event[N]的名称,发生时间的时、分、秒

计算 event[N]发生时间的秒数

当输入非全 0 时循环

　　N＋＋

　　输入 event[N]的名称,发生时间的时、分、秒

　　计算 event[N]发生时间的秒数

按事件的秒数用冒泡排序法对事件进行排序

对 i=0,…,N-1

　　显示事件的信息

　　如果 i!=0

　　　　计算与上一事件之间的时间差

　　　　将时间差转换为时、分、秒进行显示

结束

**【编程提示】**

(1) 结构体的定义结构为:

```
struct <结构体名>
{
    <成员声明的列表>
};
```

(2) 结构体变量的成员通过“.”访问,结构体变量之间可以整体赋值。结构体变量不能整体输入、输出和比较。

**【测试指南】** 输入的事件应无序,时间包括时、分、秒。

14. 某单位对职员的级别进行评定,请来专家对参评的职员打分,打分从高到低设定为 3、2、1,编程统计每个人的平均分。设参加评选的人数不超过 20。要求开始先设定参评人员的人数、姓名、编号,然后按编号的顺序输入每个专家的打分结果,第 1 个分数为 0 表示结束。按平均分从高到低显示打分结果,显示姓名、编号和平均分。参评人员信息要用结构体。

**专家打分表**

| 参评人员编号 | 评　　分 |
|---|---|
| | |
| | |

**【问题分析】** 参评人员的信息包括:姓名、编号、平均分。先定义参评人员的结构体,用结构体声明大小为 20 的数组,就可以容纳 20 个人的信息了。

设定具体人数 N 之后,每组输入的是按编号排序的 N 个分数,这是一个专家为所有参评人员打的分数。由于是给不同人的评分,还不能计算平均分,要先保存起来。每人的“平均分”成员可存放累计分数,输入完后除以专家数就是平均分。

输入全 0 时表示这是结束标志(不是对参评人的评分了)。这时确定了专家人数,每个参评人的分数个数就确定了,总分除以专家数就是平均分。排序、显示即可。

**【算法描述】** 设参评人员的结构体数组为 clerk,参评人数用 N 表示,专家人数用 M 表示。暂存分数的数组为 S。

输入 N

输入 N 个人的姓名和编号

按编号对参评人排序

M=0
输入 N 个分数到 S 中
当 S[0]不为 0 时循环
　　M++
　　将 S 中的分数分别加到 clerk 每个人的“评分”中(用循环)
　　再输入 N 个分数到 S 中
对 i=0,…,N-1
　　计算每个人的平均分
按平均分使用冒泡排序法对参评人员排序
按平均分的多少顺序输出每个人的信息
结束

## 3.5 习题 5 问题的模块化求解

1. 实现函数 int index(char t[], char s[]),用于确定字符串 t 是不是 s 的子串。若是,返回子串 t 在 s 中第一次出现时的第一个字符的下标;若不是,返回 -1。编写主函数,调用该函数查找子串。

**【问题分析】** t 在 s 中出现的下标位置是 t 的第一个字符在 s 中出现的下标位置。

**【算法描述】** t 是不是 s 的子串,即判断 t 在 s 中出是否出现,从 s 的开头逐个与 t 的每一个字符进行比较。如果全部相等,则返回 s 的当前起始位置(需要提前使用一个变量保存);如果有一个字符不相等,则继续从 s 的下一个字符开始逐个与 t 的每一个字符串比较。此时需要以下两重循环,外循环范围是 s 的全部字符,内循环范围是 t 的全部字符。千万注意,内循环中不得修改外循环变量的值。

**【编程提示】** 使用以下循环格式来进行:

```
for(int i=0;s[i]!='\0';i++)
{
    for(int j=0;t[j]!='\0';j++)
    {
        ……
    }
}
```

**【测试指南】** 应至少输入三组 t 和 s 字符串的组合,t 在 s 中出现,t 不在 s 中出现,t 比 s 长。

**【问题扩展】** 如何计算 t 在 s 中出现的次数?

2. 编写将字符串中所有小写字母转换为大写字母的函数。

**【算法描述】** 首先查表得知小写字母与对应的大写字母的距离是 32 或 97-65 或'a'-'A',比如,'M'='m'-32,知道这一点很重要。然后对字符串中的每一小写字母(需要通过'a'<=c<='z'条件判断是小写字母)进行类似的运算,即可得到对应的大写字母。

**【编程提示】** 函数声明格式如下：void lower2upper( char a[])；

**【测试指南】** 输入的字符串为：只有大写字母，只有小写字母，大小写、数字、标点混合的字符串。

**【问题扩展】** 编写将字符串中所有大写字母转换为小写字母的函数。

3. 编写函数，绘制由指定符号组成的、指定行数的如下形式的三角形，参数缺省时行数为3，字符为'*'。三角形的形式如下：

```
    &
   &&
  &  &
 &    &
&&&&&&&&&
```

这是由'&'组成的5行的三角形。在主函数中输入行数和字符，调用该函数绘制三角形，缺省某些参数调用该函数绘制三角形。

**【问题分析】** 本题的关键是确定每行起始符号前的空格数。设三角形的行数为n，则第i行的字符数为2i－1，i＝1，…，n。这样就可算出第1个字符前的空格数为(2n－1－(2i－1))/2，中间的空格数为(2i－1－2)。

**【算法描述】** 设组成三角形的符号为c。

打印(2n－1－1)/2个空格

打印c

对i＝2，…，n－1

　　打印(2n－1－(2i－1))/2个空格

　　打印c

　　打印(2i－1－2)个空格

　　再打印c，换行

打印2n－1个c，换行

结束

**【编程提示】** 打印三角形需要知道三角形的函数和组成三角形的字符，这就是函数的两个参数，本函数不需要返回值。打印若干个空格的功能最好也写成一个函数，参数是空格的个数，返回值是指向有空格组成的字符串的指针。存放空格的字符数组声明为静态变量。特别注意空格组成的字符串末尾存放'\0'。

**【问题扩展】** 除了打印三角形外，还可以尝试打印平行四边形、菱形、圆、椭圆、房屋、文字，可以选择实心或空心等，还可以绘制正弦曲线、余弦曲线，这些曲线甚至可是不同频率和幅值的等。

4. 编写函数，求两个数的最大公约数。

**【算法描述】** 首先对传递过来的两个整数进行交换，以保证大的在前，小的在后。然后使用辗转相除法：求两个数的余数，并把除数和余数作为新的两个数(待求最大公因数的两个数)，循环进行辗转相除，直到余数为零，此时的除数即为所求的最大公约数。

**【编程提示】** 函数声明格式如下：

```
int   gcd(int p, int q);
```

**【问题扩展】** 编写函数,求两个数的最小公倍数。

5. 求出200～1000之间所有这样的整数,它们的各位数字之和等于5,其中判断一个数的各位数字之和是否等于5的功能应写为一个函数。

**【算法描述】** 由于是三位和四位十进制数,所以只需要取得数的个、十、百、千位,然后求和,最后判断是否为5即可。公式如下:

```
a=i%10;                //个位
i/=10;
b=i%10;                //十位
i/=10;
c=i%10;                //百位
i/=10;
d=i%10;                //千位
```

**【编程提示】** 函数声明格式如下:

```
bool   IsSum5(int x);
```

6. 编写程序计算 p=n!/(r!(n−r)!) (n>r),其中阶乘的计算写成函数。

**【算法描述】** 有两种计算阶乘的算法。

一种是循环从1开始逐步加1,循环如下:

```
for(int i=1; i<=n; i++)
```

另一种是循环从n开始逐步减1,循环如下:

```
for(int i=n; i>=1; i--)
```

也可以使用以下循环:

```
while(n--)
```

**【编程提示】** 阶乘函数声明格式如下:

```
double   fact(int n);
```

7. 从键盘上输入一个大于4的整数,然后将从4开始到该数之间的所有整数分解为两个素数之和,显示出每个整数的分解情况,例如:4=2+2,6=3+3,8=3+5等。

**【算法描述】** 首先编写判断一个数是否为素数的函数,函数声明格式如下:

```
bool prime(int m);
```

判断素数的算法是,将小于该数的全部整数(从2开始)逐个去除该数,求余数。如果余数都不为零,说明是素数;否则不是。

然后将该数分解为两个数之和,判断这两个数是否都是素数,是则显示结果,否则重新分解。

**【编程提示】** 函数声明格式如下:

```
void  decompose (int n);
```

8. 编写函数,用选择法对 10 个数进行从小到大的排序。

**【问题分析】** 将数据序列分为两部分,一个是已排好的部分(开始为空),另一个是待排的部分(开始为全部数组元素)。

从待排序部分中选出最小的一个元素,放在已排好的部分的最后,循环进行直到全部待排序部分中的元素用完为止。

**【算法描述】**

(1) 从有序数列{}和无序数列$\{a_0, a_1, \cdots, a_{n-1}\}$开始进行排序。

(2) 处理第 i 个元素时($i=0, 2, \cdots, n-2$),数列$\{a_0, a_1, \cdots, a_{i-1}\}$是已有序的,而数列$\{a_i, a_{i+1}, \cdots, a_{n-1}\}$是无序的。对无序区元素均与其第一个元素(无序部分的)进行比较后从中选择一个最小元素 $a_k$放入有序区的尾部。

(3) 重复第(2)步,共进行 n−1 次操作。

**【编程提示】** 选择法排序函数声明格式如下:

```
void  SelectSort(int a[], int n);
```

算法中需要两重循环,结构如下:

```
for(int i=0; i<n-1; i++)
{
    for(int j=i+1; j<n; j++)
    {
            ......
    }
}
```

9. 写一个判断素数的函数,在主调函数中输入一个整数后,由该函数输出是否是素数的信息。

10. 编写函数 fun,它的功能是:计算正整数 n 除 1 和 n 之外的所有因子之和,并返回此值。

**【算法描述】** 从 2 到 n−1 循环除 n 求余。如果余数为零,则将循环变量加入和中,并将循环变量减 1(避免遗漏掉重复的因子),继续循环;余数不为零时,继续循环。

**【编程提示】** 函数声明格式如下:

```
int  Sumfactor(int n);
```

11. 编写函数 SeriesSum(),它的功能是:计算下列级数的和,并返回此值。

$S=1+x+x^2/2!+x^3/3!+\cdots+x^n/n!$,其中 n 和 x 由键盘输入。

**【算法描述】** 记该级数的通项为 $u_n$,则 $u_n=x^n/n!$,$u_{n-1}=x^{n-1}/(n-1)!$,从而,

$$u_n=u_{n-1} * x/n$$

使用以上递推公式,可以大大减少运算次数。

接下来就是准备变量并组织循环如下:

```
for(int i=1;i<=n;i++)
{
  u=u* x/i;
  S+=u;
}
```

**【编程提示】** 函数声明格式如下：

```
double  Series Sum(int n, double x);
```

12. 编写函数 fun,它的功能是：计算 1～n 之间能同时被 3、5 和 7 整除的所有自然数之和,并返回此值。

**【问题分析】** 能同时被 3、5 和 7 整除的条件是：

```
i%3==0  &&  i%5==0  &&  i%7==0
```

逐个判断即可。

13. 从键盘输入两个整数 m 和 n,然后从 m+1 开始找出大于 m 的 n 个素数。

**【问题分析】** 本题仍然需要定义判断素数的函数,然后从 m+1 开始找到 n 个素数。函数声明格式如下：

```
void fun(int m,int n);
```

**【编程提示】** 函数体中的循环如下：

```
i=m+1;
while(n)
{
    if(prime(i))        //判断素数
    {
        显示 i
        n--;
    }
    i++;
}
```

14. 从键盘输入 10 个字符串,找出其中最大者并输出,假定每个字符串长度不超过 80 个字符。

**【问题分析】** 字符串最大者指按字母顺序排序,排在最后面的单词。

**【算法描述】** 首先搞清楚如何定义字符串数组,定义格式为：char str[10][81];它可以表示 10 个字符串,每个字符串的长度不超过 80。C++ 语言提供库函数 strcmp 可以用来比较两个字符串的大小,strcpy 用于字符串的赋值,但本题要求自己编写这样的函数。

**【编程提示】** 函数声明为：

```
void  max( int str[][81],int  N);                    //N表示字符串的个数
```

15. 用牛顿迭代法(简称牛顿法)求方程 $2x^3-4x^2+3x-6=0$ 在1.5附近的根。

**【问题分析】** 设 $x_0$ 是 $f(x)=0$ 近似根,将 $f(x)$ 在 $x_0$ 附近作 Taylor 展开:

$$f(x)=f(x_0)+f'(x_0)(x-x_0)+f'(x_0)(x-x_0)^2/2!+\cdots\cdots$$

舍去平方项及以后的项,将 $f(x_0)+f'(x_0)(x-x_0)$ 作为 $f(x)$ 的近似,则

$$f(x_0)+f'(x_0)(x-x_0)=0$$

的根为:

$$x=x_0-f(x_0)/f'(x_0)$$

将它作为 $f(x)$ 的一个新的近似根。如果它不满足精度要求(如 $f(x)>1.0e-7$),则将其代入上式的右边可以再求出一个更好的近似根,直到满足精度要求。这就是解一次非线性方程的牛顿(Newton)迭代公式(牛顿法):

$$x_{n+1}=x_n-\frac{f(x_n)}{f'(x_n)}$$

其中,$f'(x_n)$ 是 $f(x)$ 在 $x_n$ 处的导数。

**【算法描述】** 首先编写两个函数,f(x)和 f1(x),然后在主函数中对 x 赋初值1.5,通过以上迭代公式不断求根,直到两次求根的差为1.0E-7为止。

**【编程提示】** 函数声明格式如下:

```
double f(double x);            //待求根的函数
double f1(double x);           //导函数
```

也可以将牛顿迭代法编写为一个函数,由主函数调用。函数声明格式如下:

```
double root(double (*f)(double), double (*f1)(double), double initvalue, double delta);
```

在主函数中的调用格式如下:

```
y=root(f,f1,1.5,1.0E-7);
```

16. 用弦截法求一元非线性方程 $f(x)=xe^x-1=0$ 在区间[0.5,0.6]中的根。

**【问题分析】** 设方程 $f(x)=0$ 在区间 $[x_0,x_1]$ 中有单根,且 $f(x_0)$ 和 $f(x_1)$ 异号,则过两点 $(x_0,f(x_0))$、$(x_1,f(x_1))$ 的直线方程为:

$$\frac{x-x_0}{y-f(x_0)}=\frac{x_0-x_1}{f(x_0)-f(x_1)}$$

如果用该直线作为原 f(x)的近似,用该方程的根作为原方程的根,解出新的近似根为:

$$x_2=x_0-\frac{x_0-x_1}{f(x_0)-f(x_1)}f(x_0)$$

若 $f(x_0)f(x_2)>0$ 则新的求根区间为 $[x_2,x_1]$,若 $f(x_1)f(x_2)>0$,则新的求根区间为 $[x_0,x_2]$,再用上述方法求解新的近似根,直到新的近似根 $x_n$ 满足 $|f(x_n)|<\varepsilon$。$\varepsilon$ 可取1.0E-8。

**【问题分析】** 编写待求根的函数 f(x),然后在主函数中对 x 分别赋0.5和0.6,通过以上迭代公式不断求根,直到两次求根的差为1.0E-7为止。

**【编程提示】** 函数声明格式如下:

```
double f(double x);            //待求根的函数
```

也可以将弦截法编写为一个函数，由主函数调用。函数声明格式如下：

```
double root(double (* f)(double), double x0, double x1, double delta);
```

该函数的迭代部分应处理函数在两个点的值同号和异号问题，并改变迭代区间。

在主函数中的调用格式如下：

```
y=root(f, 0.5, 0.6, 1.0E-7);
```

**【问题扩展】** 用弦截法求方程 $x^3-5x^2+16x-80=0$ 在(2,6)之间的根。

17. 使用梯形法计算定积分 $\int_a^b f(x)dx$ 的值，其中 $a=0,b=1,f(x)=\sin(x)$。

**【问题分析】** 将积分区间分成 n 等份，每份的宽度为 $(b-a)/n=h$，在区间 $[a+i\cdot h, a+(i+1)h]$ 上使用梯形的面积近似原函数的积分。则：

$$\int_a^b f(x) = \sum_{i=0}^{n-1}\int_{a+ih}^{a+(i+1)h} f(x) \approx \sum_{i=0}^{n-1}\frac{h}{2}(f(a+ih)+f(a+(i+1)h))$$
$$= h\left(\frac{f(a)+f(b)}{2}+\sum_{i=1}^{n-1}f(a+ih)\right)$$

这就是数值积分的梯形求积公式。n 越大或 h 越小，积分就越精确。本题 n 可以取 1000，或让 h 是一个较小的值。

**【算法描述】** 首先编写一个函数，f(x)，然后在主函数中对 x 分别赋 0 和 1，通过以上迭代公式不断求根，直到两次求根的差为 1.0E−7 为止。

**【编程提示】** 函数声明格式如下：

```
double f(double x);
```

也可以将弦截法编写为一个函数，由主函数调用。函数声明格式如下：

```
double root(double (* f)(double), double a,double b,double delta);
```

该函数的迭代部分应处理函数在两个点的值的同号和异号问题，并改变迭代区间。

在主函数中的调用格式如下：

```
y=root(f,0,1,1.0E-7);
```

18. 重载求三个数中最大值的函数。三个数的类型可能是整型、实型(float)、双精度型和字符型。

**【编程提示】** 这三个重载函数的声明格式如下：

```
int      max(int   x,  int y,  int  z);
float    max(float x,  float y,  float z);
double   max(double x,  double y,  double z);
char     max(char x,  char y,  char z);
```

# 3.6　习题 6　按址操作——指针

本章的所有题目都要求使用指针来编写。

1. 输入 10 个字符串，找出最大的字符串，要求使用指针数组实现。

**【问题分析】** 先将第一个串设为最大串，而后向后依次用当前已找到的最大串和后续串比较，并始终保留当前已找到的最大串的指针。比较两个字符串的大小可以利用 strcmp 函数来实现。

**【编程指导】**

```
//程序框架
char   * ptr[10], * q;
循环 10次 {
    ptr[i]=new char[100];                    //分配长度为 100的空间
    cin>>prt[i];                             //输入一个字符串
}
q=ptr[0];
从 1-9循环 9次 {
    if(字符串 ptr[i] 大于 字符串 q)          //利用 strcmp(),或自己编写比较函数
        q=ptr[i];
}
```

输出最大字符串 q 的信息；

**【结果示例】**

```
输入：abc bcd cde def efg fgh ghi hij ijk xy
输出：xy
```

2. 从键盘输入 10 个字符串，假定每个字符串长度不超过 80 个字符，然后对这 10 个字符串进行排序，最后输出排序后的结果。

**【问题分析】** 可以用二维数组存储用户输入的 10 个字符串，而后用某种排序法对其排序(下面的编程指导中采用了选择排序)。在排序过程中，可以用指针操作数组，实现字符串比较大小、交换等操作。

**【编程指导】**

```
//程序框架
char   str[10][80];
循环 10次 {
    cin>>str[i];                             //输入
}
// 选择排序
循环 i=0,…,8                                 //每次选择的"最大"元素与 str[i]互换
    k=i                                      //设当前最小的元素的下标保存在 k中
    循环 j=i+1,…,9                           //与后面的所有元素比较
```

```
        若 字符串(str+j)<字符串(str+k),则 //利用 strcmp 函数实现
                k=j                       //记下最小元素的下标
    如果 k!=i                              //str[i]不是最小的元素
        *(str+i) 和 *(str+k) 交换          //交换最小元素和 str[i]
输出 10个字符串;
```

**【结果示例】**

输入:efg bcd cde fgh xy abc def ghi ijk hij
输出:abc bcd cde def efg fgh ghi hij ijk xy

3. 编写函数,求出一个字符串的长度,要求使用地址传递。

**【问题分析】** 要利用函数求出字符串的长度,则该函数必须有接受一个字符串的参数。因此函数的形式参数可以是 char 类型的数组或 char 类型的指针。这里采用 char 类型指针作为形式参数。另外,字符串的尾部一定是'\0'。利用这一特点就可求出字符串的长度。

**【编程指导】**

```
//定义函数
int myStrlen(char * p)
{
    循环读取字符串 p的内容,直到字符 '\0'结束,同时计算串长度;
    返回字符串长度;
}
```

在主函数中输入一个串,利用 myStrlen 计算其长度。

**【结果示例】**

输入:abcd1234
输出:8

4. 编写函数,将一个字符串中指定的字符删去,然后输出新的字符串。

**【问题分析】** 首先,函数从一个字符串中删除字符,则该函数应有两个参数,一个是原始串的指针,另一个是待删除的字符。

另外,在一个字符串中删除某个字符,可采用如下方法:从前至后判定某个位置的字符是否要删除,若要删除,则将该位置后面的所有元素都前移一个存储位置。

**【编程指导】**

```
//定义删去字符的函数
void myDelChar(char * p,char c)
{
    while(* p!= '\0')
    {
        if(* p==c) {
            char * q=p;
            while(* q != '\0')
```

```
            {
                * q= * (q+1);   q++;
            }
        }else {  p++;}
    }
}
```

在主函数中输入一个串,利用 myDelChar 删除某个字符,将结果输出。

【结果示例】

```
输入:SCAHAOOLA  A
输出:SCHOOL
```

5. 用指针数组保存 12 个月份的英文名称,输入一个月份后,显示该月的英文名称,例如,输入 1,则显示"January";如果输入的月份值不在 1～12 之间,则显示"Input Error"信息。

【本题目的】 练习使用字符串指针操作字符串。

【编程指导】 首先可在主函数定义以下指针数组:

```
char * pName[12]={"January","February","March","April","May", "June",
    "July", "August","September","October","November","December" };
```

而后根据用户输入的整数 k,输出 pName[k－1]即可。当然,如果输入的月份值不在 1～12 之间,则提示输入错误。

【结果示例】

```
输入:2
输出:February
```

6. 编写函数,将一个字符串中所有的大写字母转换为小写字母,所有的小写字母转换为大写字母,函数调用时使用地址传递。

【问题分析】 这个问题有两个要点。

第一,为了向函数传递字符串,应该以 char 类型指针作为函数的形式参数。

第二,逐个检查每个字符,将其中大写字母转换为小写字母。转换方式为:

```
ch= ch- 'A'+ 'a';                                //大写转小写
```

【编程指导】

```
//转换函数
char * tran(char * ch)
{
    循环逐个检查每个字符,将其中大写字母转换为小写字母;
    返回指针 ch;                                 //注意,ch 的值在函数中不要改变
}
int main()
{
```

```
    定义 char 类型数组 str[100];                //长度自己定
    向 str 读入信息;
    tran(str);
    输出 str;
}
```

**【结果示例】**

输入：ABcdEfXyz
输出：abcdefxyz

**【思考问题】**

在编程指导中给出的转换函数有一个 char 指针类型的返回值，该返回值与传入的 char 指针相同，这说明转换函数的返回值实际上就是传入的字符串的指针。如果这个返回值设为 void，那么函数是否仍然起作用？设置这个返回值的好处是什么？

7. 编写函数，分别统计一个字符串中的大写字母、小写字母、数字字符和其他字符的个数。

**【问题分析】** 为了向函数传递字符串，应该以 char 类型指针作为函数的形式参数。另外，为了从函数中得到各类型字符的数量，可以以指针方式传入四个整型变量，用来取回统计结果。

统计方法是逐个检查每个字符，判别每个字符的类型（大写字母、小写字母等）并加以统计。

**【编程指导】**

```
//统计函数
void statistic(char * str, int * Up, int * Low, int * Digit, int * Other)
{
    将 Up, Low, Digit, Other 都置为 0;
    循环逐个检查 str 每个字符{
        将大写字母个数累加到 Up;
        将小写字母个数累加到 Low;
        将数字个数累加到 Digit;
        将其他字母个数累加到 Other;
    }
}
int main()
{
    int Up, Low, Digit, Other;
    定义 char 类型数组 str[100];                //长度自己定
    向 str 读入信息;
    statistic(str,&Up, &Low, &Digit, &Other);
    输出 str;
}
```

**【结果示例】**

输入：QWEcdfy1123@#

输出：大写字母 3 小写字母 4 数字字符 4 其他字符 2

8. 使用多级指针找出多个字符串中最大的一个(以 ASCII 码值为依据)。

**【本题目的】** 学习多级指针的使用方式。

**【问题分析】** 利用二级指针指向指针数组，通过二级指针的前后移动，可以遍历指针数组所指向的每一个字符串，从而判定大小。

可先将第一个串设为最大串，而后向后依次用当前已找到的最大串和后续串比较，并始终保留当前已找到的最大串指针即可。比较两个字符串的大小可以利用 strcmp 函数来实现。

**【编程指导】**

```
//程序框架
int main()
{
    char * q[4]={"abc","123","xyz","ijk"};     //定义指向字符串的指针数组
    char **p=q;                                //二级指针指向指针数组
    char * pMax=q[0];                          //设 pMax 指向最大字符串
    while(p<=&q[3]) {
        if(strcmp(pMax, * p)<0)  pMax= * p;
        p++;
    }
    cout<<pMax;
}
```

**【结果示例】**

输出：xyz

9. 使用指针数组对多个字符串进行排序。

**【问题分析】** 可以定义指针数组并为每个指针设定初始字符串，而后用某种排序法对其排序。字符串比较大小可以用 strcmp 实现，字符串交换时交换两者指针即可。

**【编程指导】**

```
//程序框架
char * p[10]={"efg","bcd","cde","fgh","xy","abc","def","ghi","ijk", "hij" };
//选择排序
循环 i=0,…,8                           //每次选择的"最大"元素与 p[i]互换
    k=i                                //设当前最小的元素的下标保存在 k 中
    循环 j=i+1,…,9                     //与后面的所有元素比较
        若 字符串 p[j]<字符串 p[k],则   //利用 strcmp 函数实现
            k=j;                       //记下最小元素的下标
    如果 k!=i                          //p[i]不是最小的元素
```

```
        char * q;
        q=p[i];
        p[i]=p[k];                          //交换地址
        p[k]=q;
输出 10个字符串;
```

**【结果示例】**

输出：abc bcd cde def efg fgh ghi hij ijk xy

10. 在主函数中定义并初始化一个一维数组,在被调函数中计算该一维数组中的元素之和并将结果返回给主函数。

11. 已知有 5 个学生各有 4 门课程成绩,完成以下要求：

(1) 计算每一门课程的最高分。

(2) 计算每个学生的各门课程平均分。

(3) 分别统计每个同学不及格课程的门数。

(4) 输出平均成绩在 90 分以上的学生的信息。

(5) 输出每门课程都在 85 分以上的学生的全部成绩和平均成绩。

以上每个要求都分别通过不同的函数实现。

**【注】** 本题参见 2.6.6 节的实验指导。

12. 输入一个 32 位二进制写法的 IP 地址,然后判断该地址属于 A、B、C、D、E 中的哪一类。

**【问题分析】** IP 地址的分类方法是：

A 类：二进制的 IP 地址首位为 0,即第一位为 0。

B 类：二进制的 IP 地址前两位为 10,即第二位为 0。

C 类：二进制的 IP 地址前三位为 110,即第三位为 0。

D 类：二进制的 IP 地址前四位为 1110,即第四位为 0。

E 类：二进制的 IP 地址前五位为 11110,即第五位为 0。

所以只要判断二进制写法第几位为 0 即可。

**【编程指导】**

```
//程序主体部分示意
int i=0;
char * str[]={"A类","B类","C类","D类","E类"},ip[40];
cin>>ip;
char * p=ip;
while(i<5)
{
    若(* p==0) 则 IP地址属于 str[i],结束循环;
    p++;  i++;
}
```

**【结果示例】**

输入：11000000110000001000000011000000111
输出：C类

13. 输入一个点分十进制写法的 IP 地址，然后将其转换为 32 位的二进制写法。

**【注】** 本题目参见 2.6.5 节的实验指导。

14. 定义一个结构体变量，包括年、月、日，编程计算该日在本年中是第几天，在计算时要注意闰年的问题。

**【本题目的】** 熟悉利用指针访问数组的方法，熟悉利用指针操作结构体的方法。

**【问题分析】** 首先要定义包括年、月、日的结构体类型，定义包含各个月份天数的整型数组。

可以定义指针分别指向结构体对象和包含各个月份天数的数组。

接受用户输入的数据后，应当判断年份是否是闰年。若是闰年，则应修正二月份的天数。

计算从 1 月到当前月前一个月份的天数之和，再加上当前月的天数，就是所要求的结果。

判断闰年的规则为：

(1) 能被 4 整除且不能被 100 整除的为闰年。(如 2004 年是，1900 年不是。)

(2) 能被 400 整除的是闰年。(如 2000 年是，1900 年不是。)

**【编程指导】**

```
//定义日期结构体
struct Date {
    int year, month, day;
};
//主函数框架
int main()
{
    //定义各个月份的天数
    int NumOfMonth[13]= {0,31,28,31,30,31,30,31,31,30,31,30,31};
    int * pNum= NumOfMonth;                    // 指针指向数组 NumOfMonth
    定义结构体对象 date;
    用户从屏幕输入对象 date 的各个分量，即 date.year, date.month, date.day;
    定义结构体指针 p,令其指向对象 date;
    if(p- > year 是闰年)
        NumOfMonth[2]= 29;
    存放累加和的 sum 置为 0;
    for(int i 从 1 到 p- > month- 1)           //从 1月到当前月前一个月份
        累加 pNum[i] 到 sum;
    将最后一个月需要累加的天数 date.day 累加到 sum;
    输入 sum;
}
```

【结果示例】

输入：2013 3 31
输出：90

## 3.7 习题7 数据的抽象与封装——类

1. 定义并实现一个电子钟类E_Clock。该类包括的特征信息有：Hour(时)、Minute(分)、Second(秒)以及不带参数的构造函数(将时间初始化为0时0分0秒)、带参数的构造函数E_Clock (int,int,int)(将时间初始化为当前系统的时间值)、显示时间函数ShowTime、增加1秒钟函数AddSecond,要求该电子钟具有计时和电子表的功能。

**【问题分析】** 本题的功能可以自己设计。模拟电子表,通常功能有:设置时间、获取系统时间、报时、倒计时、正计时、走时。注意由于时间是无限的,如果用无限循环,一旦选择走时,便无法使用其他功能了。所以,走时功能可设为走时10秒或20秒,不要无限走下去。

**【算法描述】** 类的成员变量全部为私有,类的成员函数(包括构造函数)全部为公有。将时、分、秒设计为三个整数类型的成员变量,设计两种重载的构造函数,一个不带参数,另一个带三个参数,显示时间函数和增加1秒钟的函数均为无返回值、无形参的函数。

**【编程提示】** 类的框架格式如下:

```
class E_Clock
{
private:
  int Hour;
  int Minute;
  int Second;
public:
  E_Clock();
  E_Clock(int Hour,int Minute,int Second);
public:
  void ShowTime();
  void AddSecond();
};
```

下面的程序可以创建一个电子时钟对象EC,并初始化为当前系统的时间。执行time(long &t);后,可获取系统的当前时间,其中t为long变量,存放距1970年1月1日的秒数(GMT时间)。库函数time(long *)在头文件ctime中声明,因此需包含#include <ctime>命令。

```
time(&t);                                    //距1970年1月1日的秒数
h=t/3600;
m=(t-h*3600)/60;
```

```
s=t%60;
h=(h+8)%24;                                  //北京与 GMT 的时差为 8
E_Clock EC(h,m,s);
```

执行库函数 Sleep(1000)可以模拟延时 1 秒的时间,即 Sleep(unsigned long),其中 Sleep 的参数以毫秒为单位,需要包含＃include <windows.h>。

2. 定义并实现一个有理数类 Rational,该类包括特征信息有:分子 numberator、分母 denominator 以及构造函数、两个有理数相加函数 RationalAdd、相减函数 RationalSub、相乘函数 RationalMul、相除函数 RationalDiv、以分子/分母形式输出函数 RationalPrint、化简分数函数 Fsd、求最大公约数函数 Gcd,要求对两个类对象进行加、减、乘、除并进行化简输出。

**【算法描述】** 类的成员变量全部为私有,类的成员函数(包括构造函数)全部为公有。将分子、分母设计为两个整数类型的成员变量。设计两种构造函数的重载,一个不带参数,另一个带两个参数。相加、相减、相乘和相除函数的返回类型均为有理数类、一个形参,类型为一个有理数类。输出函数和化简分数函数无返回类型,无形参。最大公约数函数返回类型为整数,无形参。

**【编程提示】** 类的框架格式如下:

```
class Rational
{
private:
  int numberator;
  int denominator;
public:
  Rational();
  Rational(int numberator,int denominator);
public:
  Rational Add(Rational r2);
  Rational Sub(Rational r2);
  Rational Mul(Rational r2);
  Rational Div(Rational r2);
public:
  void Print();
  void Fsd();
  int Gcd();
};
```

3. 定义并实现一个公民类 Citizen,该类包括的特征信息有:身份证号 id、姓名 name、性别 gende、年龄 age、籍贯 birthplace、家庭住址 familyAddress 等属性以及构造函数、输入公民信息函数 input 以及输出公民信息函数 output,要求能够对该类对象进行初始化、输入和输出操作。

**【算法描述】** 类的成员变量全部为私有,类的成员函数(包括构造函数)全部为公有。将身份证号、年龄分别设计为整数类型的成员变量,将性别设计为字符类型的成员变量,将

姓名、籍贯和住址分别设计为字符串类型的成员变量。设计两种构造函数的重载，一个不带参数，另一个带 6 个参数。输入公民信息函数和输出公民信息函数无返回类型，无形参。

**【编程提示】** 类的框架格式如下：

```
class Citizen
{
private:
   int id;
   char name[20];
   char gende;
   int age;
   char birthplace[20];
   char familyAddress[20];
public:
   Citizen();
   Citizen(int id,  char name[],  char gende,  int age,
           char birthplace[],  char familyAddress[]);
public:
   void input();
   void output();
};
```

4. 定义并实现 Dog 类，包含 name、age、sex、weight 等属性以及初始化和显示属性的方法，要求用一般成员函数和构造函数两种方法实现初始化操作。

**【算法描述】** 类的成员变量全部为私有，类的成员函数（包括构造函数）全部为公有。将名字设计为字符串类型成员变量，将年龄和体重设计为整数类型成员变量，将性别设计为字符类型成员变量。设计两种构造函数的重载，一个不带参数，另一个带四个参数。初始化函数带四个参数。显示属性函数无返回类型，无形参。

**【编程提示】** 类的框架格式如下：

```
class Dog
{
private:
   char name[20];
   int age;
   char sex;
   int weight;
public:
   Dog();
   Dog(char name[],  int age,  char sex,  int weight);
public:
   void init(char name[],  int age,  char sex,  int weight);
   void show();
};
```

5. 定义并实现 Circle 类,采用左上角和右下角坐标表示圆,具有计算面积和周长等函数,要求使用构造函数初始化。

**【问题分析】** 构造函数是函数名与类名相同的成员函数,可以重载,在声明对象时自动调用。声明对象时根据对象后有无括号和括号中的参数不同调用相应的构造函数。

**【编程提示】** 类的成员变量全部为私有,类的成员函数(包括构造函数)全部为公有。将左上角和右下角坐标设计为四个整数类型成员变量。设计两种构造函数的重载,一个不带参数,另一个带四个参数。计算面积函数和周长函数无形参,返回类型为浮点数。

类的框架格式如下:

```
class Circle
{
private:
  int x1,y1;
  int x2,y2;
public:
  Circle();
  Circle(int x1,int y1,int x2,int y2);
public:
  double area();
  double length();
};
```

6. 定义并实现三角形类,其成员变量包括三个边长变量,成员函数包括判断是否合法、计算面积,以及是否构成直角三角形、锐角三角形的钝角三角形等函数。

**【编程提示】** 类的成员变量全部为私有,类的成员函数(包括构造函数)全部为公有。将三个边长为三个浮点类型成员变量。设计两种构造函数的重载,一个不带参数,另一个带三个参数。判断是否合法函数和是否构成直角三角形、锐角三角形和钝角三角形等函数无形参,返回类型为布尔。计算面积函数无形参,返回类型为浮点数。

类的框架格式如下:

```
class Triangle
{
private:
  double a,b,c;
public:
  Triangle();
  Triangle(int a,int b,int c);
public:
  bool IsTriangle();
  double area();
  bool IsTriangle1();
  bool IsTriangle2();
  bool IsTriangle3();
};
```

7. 定义并实现 Time 类,包括设置时间、进行时间的加减、按照各种可能的格式输出时间,要求设计多个重载的构造函数。

**【编程提示】** 类的成员变量全部为私有,类的成员函数(包括构造函数)全部为公有。将时、分、秒设计为三个整数类型成员变量。设计四种构造函数的重载,一个不带参数,另三个分别带一个、两个和三个参数。设置时间函数为无返回值、无形参的函数。时间加减函数返回类型为时间类。它有一个形参,类型也是时间类。格式输出函数为无返回值、无形参的函数。

类的框架格式如下:

```
class Time
{
private:
   int Hour;
   int Minute;
   int Second;
public:
   Time();
   Time(int Hour);
   Time(int Hour,int Minute);
   Time(int Hour,int Minute,int Second);
public:
   void SetTime(int Hour,int Minute,int Second);
   Time Add(Time t2);
   Time Sub(Time t2);
   void PrintTime1();
   void PrintTime2();
   void PrintTime3();
   void PrintTime4();
};
```

8. 定义并实现地址类 Address,包括姓名、所居住的街道地址、城市和邮编等属性以及改变对象姓名的 Changename 函数、显示地址信息的 Display 函数。

**【编程提示】** 类的成员变量全部为私有,类的成员函数(包括构造函数)全部为公有。将姓名、所居住的街道地和城市设计为三个字符串类型成员变量,将邮编设计为一个整数类型成员变量。设计两种构造函数的重载,一个不带参数,另一个带四个参数。改变姓名函数为无返回值、形参为字符串类型的函数。显示地址信息函数为无返回值、无形参的函数。

类的框架格式如下:

```
class Address
{
private:
   char name[20];
```

```
    char street[20];
    char city[20];
    int code;
  public:
    Address();
    Address(char name[], char street[], char city[], int code);
  public:
    void Changename(char name[]);
    void Display();
  };
```

9. 定义并实现三维空间的Point3D类，包括x、y、z三个成员变量，一个计算空间中两个点之间的距离的成员函数，并编写合适的构造函数和析构函数。

**【编程提示】** 类的成员变量全部为私有，类的成员函数(包括构造函数)全部为公有。将x、y、z设计为三个整数类型成员变量。设计两种构造函数的重载，一个不带参数，另一个带三个参数。距离函数的返回类型为浮点数，形参为三维空间类。析构函数为无返回值无形参的函数。

类的框架格式如下：

```
class Point3D
{
private:
  int x,y,z;
public:
  Point3D();
  Point3D(int x,int y,int z);
public:
  double distance(Point3D p2);
  ~Point3D();
};
```

10. 定义并实现职工类Employee，包括工作部门Department、姓名Name、出生日期Birthday、职务EmpPosition、参加工作时间DateOfWork、工资Salary等属性以及注册职工信息函数Register、设置工资函数SetSalary、获取工资信息GetSalary和显示职工信息函数ShowMessage，其中DateOfWork是日期类Date的对象，Salary是工资类EmpSalary的对象，EmpPosition是Position枚举类型变量。Date类包含day、month和year等属性以及初始化和按年月日打印函数；EmpSalary包含基本工资Wage、岗位津贴Subsidy、房租Rent、电费CostOfElec、水费CostOfWater等属性以及计算实发工资的RealSum函数；Position是包括经理MANAGER、工程师ENGINEER、职员EMPLOYEE、工人WORKER等枚举值。要求通过职工类对象数组管理职工数据的输入和输出。

**【编程提示】** 以上三个类和一个枚举的框架格式如下：

```
class Date
```

```
{
private:
   int day,month,year;
public:
   Date();
   Date(int day,int month,int year);
public:
   void print();
};

class EmpSalary
{
private:
   double Wage;
   double Subsidy;
   double Rent;
   double CostOfElec;
   double CostOfWater;
public:
   EmpSalary();
   EmpSalary(double Wage,  double Subsidy,  double Rent,
   double CostOfElec,  double CostOfWater);
public:
   double RealSum();
};

enum Position{
MANAGER,ENGINEER,EMPLOYEE,WORKER
};

class Employee
{
private:
   char Department[20];
   char Name[20];
   char Birthday[20];
   Position EmpPosition;
   Date DateOfWork;
   EmpSalary Salary;
public:
   Employee();
   Employee(char Department[20],  char Name[20],
   char Birthday[20], Position EmpPosition,
   Date DateOfWork, EmpSalary Salary);
```

```
public:
  void Register(char Department[20],  char Name[20],
  char Birthday[20], Position EmpPosition,
  Date DateOfWork, EmpSalary Salary);
  void SetSalary(EmpSalary Salary);
  EmpSalary GetSalary();
  void ShowMessage();
};
```

## 3.8 习题8 取其精华 发挥优势——继承

1. 设计一个点类 Point 和其派生类彩色点类 ColorPoint。

**【编程提示】** 点类包括两个坐标值成员变量和具有两个形参的构造函数。彩色点类继承了点类,增加一个颜色成员变量,其构造函数增加一个形参,前两个参数传递给基类的构造函数来初始化。

两个类的框架格式如下:

```
class Point
{
private:
    int x,y;
public:
    Point(int x,int y);
};

class ColorPoint:public Point
{
private:
    int color;
public:
    ColorPoint(int x,int y,int color);        //该函数要初始化基类 Point 的成员
};
```

2. 设计一个 Person 类和其派生类教师 teacher,新增的属性有专业、职称和主讲课程,并为这些属性定义相应的方法。

**【编程提示】** 两个类的框架格式如下:

```
class Person
{
private:
    int id;
    char name[20];
    char gender;
```

```
public:
    Person(int id,char name[],char gender);
};

class teacher:public Person
{
    private:
    char major[20];
    char level[20];
    char course[20];
public:
    teacher(int id,char name[],char gender,char major[],
           char level[],  char course[]);
};
```

3. 设计一个汽车类 vehicle,包含的数据成员有车轮个数 wheels 和车重 weight。小车类 car 是它的私有子类,其中包含载人数 passenger_load。卡车类 truck 是 vehicle 的私有子类,其中包含载人数 passenger_load 和载重量 payload。每个类都有相关数据的输出方法。

**【编程提示】** 三个类的框架格式如下:

```
class vehicle
{
private:
  int wheels;
  int weight;
public:
  vehicle(int wheels,int weight);
};

class car:private vehicle
{
private:
  int passenger_load;
public:
  car(int wheels,int weight,int passenger_load);
};

class truck:private vehicle
{
private:
  int passenger_load;
  int payload;
public:
```

```
   car(int wheels,int weight,int passenger_load,int payload);
};
```

4．研究生类既有学生类的特征，又有教师类的特征。试通过多重继承说明一个研究生类，包括设置学生和教师的相关属性以及显示学生和教师的相关属性等功能。

**【编程提示】** 三个类的框架格式如下：

```
class 学生类                    //-------------学生类------------
{
private:
   int stdno;
   char stdname[20];
   char stdclass[20];
public:
   学生类(int stdno,  char stdname[],  char stdclass[]);
   void 设置学生属性(int stdno,  char stdname[],  char stdclass[]);
   void 显示学生属性();
};

class 教师类                    //---------------教师类-------------
{
private:
   int tchno;
   char tchname[20];
   char tchunit[20];
public:
   教师类(int tchno,  char tchname[],  char tchunit[]);
   void 设置教师属性(int tchno,  char tchname[],  char tchunit[]);
   void 显示教师属性();
};
class 研究生类:public 学生类,public 教师类
                                //-------------研究生类------------
{
private:


public:
   研究生类(int stdno,  char stdname[],  char stdclass[],
            int tchno,  char tchname[],  char tchunit[]);
   void 设置属性(int stdno,  char stdname[],  char stdclass[],
                 int tchno,  char tchname[],  char tchunit[]);
   void 显示属性();
};
```

5．日期时间类 DateTime 既有日期类 Date 的特征，又有时间类 Time 的特征，试通过多继承说明一个日期时间类，包括设置时间和日期、进行时间和日期的加减运算、按照各

种可能格式输出时间和日期等功能。

**【编程提示】** 三个类的框架格式如下：

```
class Date                          //-------------日期类-------------
{
  private:
    int year,month,day;
  public:
    Date(int year,int month,int day);
    void SetDate(int year,int month,int day);
    void ShowDate1();
    void ShowDate2();
    void ShowDate3();
};

class Time                          //-------------时间类-------------
{
  private:
    int hour,minute,second;
  public:
    Time(int hour,int minute,int second);
    void SetTime(int hour,int minute,int second);
    void ShowTime1();
    void ShowTime2();
    void ShowTime3();
};

class DateTime:public Date,public Time
                                    //-------------时间和日期类-------------
{
  private:

  public:
    DateTime(int year,int month,int day,
             int hour,int minute,int second
            );
    void Set(int hour,int minute,int second);
    void Show1();
    void Show2();
    void Show3();
};
```

6. 在几何图形类 Shape(自己设计属性和方法)的基础上,派生出椭圆类 Ellispe,其属性为圆心坐标及半长轴和半短轴的长度,并用通过构造函数对这些属性初始化,通过成

员函数计算椭圆的面积。

**【编程提示】** 两个类的框架格式如下：

```
class Shape                    //-------------形状类------------
{
  private:
    char name[20];
  public:
    Shape(char name[]);
};

class Ellispe:public Shape     //-------------椭圆类------------
{
  private:
    int x,y,r1,r2;
  public:
    Ellispe(char name[],int x,int y,int r1,int r2);
    void init(char name[],int x,int y,int r1,int r2);
    void area();
};
```

7. 用继承的方法描述下列类：商品类、家电类、电视类，自己设计其属性和方法，编写主函数对各类事物的特征和功能进行模拟。

**【编程提示】** 三个类的框架格式如下：

```
class 商品类                   //-------------商品类------------
{
  private:
    int 序号;
    char 名称[20];
    char 生产厂家[20];
    char 生产日期[20];
    char 保质期[20];
    float 定价;
  public:
    商品类(int 序号,char 名称[],char 生产厂家[],
           char 生产日期[], float 定价);
};

class 家电类:public 商品类     //-------------家电类------------
{
  private:
    int 类别;
  public:
    家电类(int 类别);
```

```
};

class 电视类:public 家电类         //-------------电视类------------
{
  private:
    int 尺寸;
    char 附件[50];
  public:
    电视类(int 序号,char 名称[],char 生产厂家[],
           char 生产日期[], float 定价,
           int 类别,int 尺寸,char 附件[]);
};
```

## 3.9 习题9 统一接口 不同实现——多态性

1. 定义一个哺乳动物Mammal类,再由此派生出狗Dog类。二者都定义Speak()成员函数,基类中定义为虚函数。定义一个Dog类的对象,调用Speak函数,观察运行结果。

**【编程提示】** 两个类的框架格式如下:

```
class Mammal                       //-------------哺乳动物类------------
{
  public:
    virtual void Speak();          //虚函数
};

class Dog:public Mammal            //-------------DOG类------------
{
  public:
    virtual void Speak();
};
```

2. 编写一个存储艺术作品的程序。艺术作品分为三类:Painting、Music和Chamber,Chamber是Music中的一类。要求如下所述:

(1) 每件作品都要标明著者、作品标题、作品诞生年份及其所属分类。

(2) Painting类要求显示画的宽和高等尺寸信息。

(3) Music类要求显示能够代表其中内容的关键信息,例如:“D Major”。

(4) Chamber类要求显示其中包括的演奏人员的数目。

**【编程提示】** 四个类的框架格式如下:

```
class 艺术作品类                    //-------------艺术作品类------------
{
  public:
```

```
    char 著者[20];
    char 作品标题[50];
    char 作品诞生年份[50];
    int 分类;
};

class Painting:public 艺术作品类    //-------------绘画作品类------------
{
  public:
    int 宽,高;
};

class Chamber                     //-------------小型器乐类------------
{
  public:
    int 演奏人员的数目;
};

class Music:public 艺术作品类      //-------------音乐类------------
{
  public:
    char 关键信息[20];
    Chamber 演奏;
};
```

3. 设计一个汽车类 Motor，该类具有可载人数、轮胎数、马力数、生产厂家和车主五个数据成员，根据 Motor 类派生出 Car 类、Bus 类和 Truck 类。其中 Bus 类除继承基类的数据成员之外，还具有表示车厢节数的数据成员 Number；Truck 类除继承基类的数据成员之外，还具有表示载重量的数据成员 Weight。每个类都有成员函数 Display，用于输出各类对象的相关信息。在主函数中分别创建各类对象，并输出各类对象的信息。

**【编程提示】**　四个类的框架格式如下：

```
class Motor                       //-------------汽车类------------
{
  private:
    可载人数、轮胎数、马力数、生产厂家和车主;
  public:
    virtual void Display();       //虚函数
};

class Car:public Motor            //-------------轿车类------------
{
  private:
  public:
```

```
    virtual void Display();

};

class Bus:public Motor              //-------------公共汽车类----------
{
  private:
    车厢节数;
  public:
    virtual void Display();
};

class Truck:public Motor            //-------------卡车类------------
{
  private:
     载重量;
  public:
    virtual void Display();
};
```

4. 定义一个 Shape 抽象类，在此基础上派生出 Square 类、Rectangle 类、Circle 类和 Trapezoid 类，四个派生类都有成员函数 CaculateArea 计算几何图形的面积，CaculatePerim 计算几何图形的周长。要求用基类指针数组，使它每一个元素指向一个派生类对象，计算并输出各自图形的面积和周长。

**【编程提示】** 五个类的框架格式如下：

```
class Shape                         //-------------形状类------------
{
  public:
    Shape();
    virtual double CaculateArea()=NULL;          //纯虚函数
    virtual double CaculatePerim()=NULL;         //纯虚函数
};

class Square:public Shape           //-------------正方形类------------
{
  private:
    double width;
  public:
    Square(double width);
    virtual double CaculateArea();
    virtual double CaculatePerim();
};
```

```
class Rectangle:public Shape          //-------------矩形类------------
{
  private:
    double width,height;
  public:
    Rectangle(double width,double height);
    virtual double CaculateArea();
    virtual double CaculatePerim();
};


class Circle:public Shape             //-------------圆类------------
{
  private:
    double r;
  public:
    Circle(double r);
    virtual double CaculateArea();
    virtual double CaculatePerim();
};

class Trapezoid:public Shape          //-------------三角形类------------
{
  private:
    double a,b,c;
  public:
    Circle(double a,double b,double c);
    virtual double CaculateArea();
    virtual double CaculatePerim();
};
```

5. 编写一个程序,分别用成员函数和友元函数重载运算符"+"和"-",实现两个矩阵的相加和相减。要求第一个矩阵的值由构造函数设置,第二个矩阵的值由键盘输入。

**【问题分析】** 友元函数是指某些虽然不是类成员却能够访问类的所有成员的函数。类授予它的友元特别的访问权。在类中,说明一个函数是友员函数,只要在函数声明的最前面写上 friend。

**【编程提示】** 类的框架格式如下:

```
class Matrix
{
private:
    int m;
    int n;
    int A[100];
```

```
public:
    Matrix();                              //默认构造函数
    Matrix(int m,int n,int B[]);           //一般构造函数
    Matrix operator+ (Matrix & ma);
                        //①成员函数形式的"+"运算符重载函数,实现对象+对象
    Matrix operator+ (int x);
                        //②另一种成员函数形式的"+"运算符重载函数,实现对象+整数
    Matrix operator- (Matrix & ma);
                        //③成员函数形式的"-"运算符重载函数,实现对象-对象
    Matrix operator- (int x);
                        //④另一种成员函数形式的"-"运算符重载函数,实现对象-整数
    void show();        //显示函数
    ~Matrix();          //析构函数

    //⑤友元函数形式的"+"运算符重载函数,实现整数+对象
    friend Matrix operator+ (int x,Matrix & ma);
    //⑥友元函数形式的"-"运算符重载函数,实现整数-对象
    friend Matrix operator- (int x, Matrix & ma);
};
```

以上框架放在程序中时请使用＃include＜iostream.h＞,并去掉 using namespace std;。友元函数的定义与非成员函数的普通函数相同(不是成员函数,不写 Matrix::)。

设 A、B、C 三个 Matrix 类的对象,根据参数类型的对应关系,C＝A＋B 将调用“＋”的重载函数①;C＝A＋3 将调用“＋”的重载函数②;C＋3＋A 将调用“＋”的重载函数⑤。对于“－”也是类似。如果没有友元函数,无法实现 C＝3＋A 形式的运算。实数和矩阵相加,可以定义为对角元是同一个实数的对角阵与矩阵相加,如 3＋A 意义是 3E＋A,其中 E 为单位矩阵。

**【问题扩展】** 如何将两个矩阵对象的加减定义为友元形式的运算符重载函数?又如何将一个矩阵对象与一个整数的加减定义为友元形式的运算符重载函数?

6. 定义字符串类 String,重载赋值运算符＝(赋值)和关系运算符＝＝(等于)、＜(小于)和＞(大于),用于字符串的赋值运算和两个字符串的等于、小于和大于的比较运算。利用 String 类实现 10 个字符串从小到大排序。

**【编程提示】** 类的框架格式如下:

```
class String
{
  private:
    char * a;
  public:
    String(char * a);
    String operator= (String & b);                    //重载运算符
    String operator== (String & b);
    String operator< (String & b);
    String operator> (String & b);
```

```
};
```

7. 重载“＝”、“＋＋”(前置和后置)和“－－”(前置和后置)运算，实现实数的加 1 以及减 1 运算。

**【问题分析】** 重载前置自增运算符 operator＋＋( )，重载后置自增运算符 operator＋＋(int)。注意 int 不是参数的类型，而仅是后置的标志。

**【编程提示】** 类的框架格式如下：

```
class Real
{
   private:
     double x;
   public:
     Real(double x);
     Real operator= (Real & b);                    //重载运算符
     void operator++ ();                           //重载前置++运算符
     void operator-- ();                           //重载前置--运算符
};
```

# 3.10　习题 10　标准输入输出与文件操作

1. 按下列格式输出圆周率的值。

```
3
3.1
3.14
3.141
3.1415
3.14159
3.141592
3.1415926
```

**【编程提示】** 利用 setprecision(n)可以设定一个实数输出时显示 n 位数字。另外，为了使用格式函数需要添加下列语句：

```
#include< iomanip>
```

2. 读取一个 C++ 源程序文件(少于 1000 行)，在每一行前面添加行号后在屏幕上输出。要求行号占 4 个字符位置，源程序文件除了右移 4 个字符外格式不变。

**【编程提示】** 为了在屏幕左侧留下 4 个字符的位置输出行号，需要使用 cout 的 width 函数进行设定。同时为保证行号一定是左对齐显示，可以使用 cout 的 setf 函数设定。另外，由于 C++ 源文件的一行可能有空格，应当利用 getline 函数读文件。

**【算法描述】**

定义 ifstream 对象并打开文件；

设行号为 1；
当(文件未处理完){
　　读取一行；
　　设定格式输出行号,行号加 1；
　　输出所读取文件内容；
}

3. 一个文本文件有多行信息,编写程序读取其内容,统计最长的一行信息和最短的一行信息各有多少字符。

**【编程提示】** 将文件的一行读入 char 类型数组,而后利用 strlen 求长度。循环找出最长和最短的行的字符数。

**【算法描述】**

定义 ifstream 对象并打开文件；
设最长字符数 max 和最短字符数 min 为 −1；
当(文件未处理完){
　　读取一行；
　　计算本行字符数存入 n；
　　若 max==−1 则 max=n,min=n；
　　否则{
　　　　若 n>max 则 max=n；
　　　　若 n<min 则 min=n；
　　}
}
输出信息；

4. 已知一个文件内容是某公司雇员的信息。每一行的内容依次是编号、姓名、籍贯、年龄。样例如下：

```
001011    刘强    上海    19
001012    王刚    陕西    28
001013    李红    四川    25
……    ……
```

编写程序,首先将文件中小于 22 岁的人依次显示在屏幕上,并计算这些人的平均年龄后输出(四舍五入到整数)。然后再将文件中籍贯为“上海”的人依次显示在屏幕上,并统计他们的人数后输出。

**【编程提示】** 本题目要对数据统计两次,一次是按年龄统计,一次是按籍贯统计。比较简单的处理方法是先后两次打开文件,分别处理每一次统计。另外,本文件的格式很明显,每一行可用三个字符串和一个整型变量读取。

**【算法描述】**

定义 ifstream 对象并打开输入文件；
定义 char 数组 ID[10]、name[16]、hometown[10]及整型变量 age；

```
当(文件未处理完){
    读取一行信息;
    若 age<22 则
        输出信息,并将 age 累加到 sum,同时累加人数到 n;
}
计算平均年龄,并四舍五入到整数后输出;
关闭文件;
再次打开该文件;
当(文件未处理完){
    读取一行信息;
    若 hometown 的值是"上海" 则
        输出信息,并累加人数;
}
输出上海籍人数;
```

5. 编写程序,实现文件复制。被复制文件和目标文件的名称由用户输入。

**【编程提示】** 读取文件信息时不要用“>>”符号,因为这个符号会抛弃空格、回车等符号。

**【算法描述】**

```
定义 ifstream 对象并以二进制方式打开输入文件;
定义 ofstream 对象并以二进制方式打开输出文件;
当(文件未处理完){
    从输入文件读取一个字节的信息;                //读到一个字符中
    若未到达文件尾部,则将该字符写入输出文件;
}
```

6. 已知一个 C++ 源程序文件,该文件包含很多注释,这些注释都由“//”引导。例如:

```
……
Student     stu2;
//打开文件
ifstream file2("file.dat",ios::binary);
if(!file2)              //处理文件打开失败的情况
{
    cout<<"文件打开失败!";
    return 1;           //结束
}
……
```

编程读取该文件,去掉注释后写入新文件 out.cpp,同时将新文件内容在屏幕上输出。

**【编程提示】** 由于符号“//”必定在一行中出现,因此可以一次读取一行,再查找有无

符号“//”。如果有注释，就将其去掉。

**【算法描述】**

定义 ifstream 对象并打开输入文件；
定义 ofstream 对象并打开输出文件；
当(文件未处理完){
　　从输入文件读取一行信息到 char 类型数组 str 中；
　　在 str 数组中查找"//"，若找到则将第一个'/'字符改为'\0'，即结束标志；
　　若未到达文件尾部，将 str 写入输出文件，同时将内容在屏幕上输出；
}

7. 一个文本文件由英文字母构成，读取该文件，将文件中的字符串“abc”换为“xyz”后写入新文件 out. txt，同时将新文件内容在屏幕上输出。

**【编程提示】** 本题目的要点是如何找字符串“abc”。一个方法是逐字符读取文件，如果读到'a'，则再读取两个字符，看看是不是“abc”。

**【算法描述】**

定义 ifstream 对象并打开输入文件；
定义 ofstream 对象并打开输出文件；
当(文件未处理完){
　　读取一个字符；
　　若未到达文件尾部，则 {
　　　　该字符若不是'a'则写入新文件；
　　　　否则{
　　　　　　再读取两个字符；
　　　　　　若后两个是“bc”，若将“xyz”写入新文件；
　　　　　　否则，将这三个字符原样写入新文件；
　　　　}
　　}
}

8. 一个文本文件中有一些正整数，这些整数用逗号分开，个数不超过 20 个。编程读取该文件，想办法得到这些整数，计算所有数字的平均值并在屏幕输出。

**【问题分析】** 由于数字之间有逗号，所以无法像有格式的文件那样直接读取整数。需要逐字符读取，而后转换为整数，并且以逗号为分割，将逗号或结束标志前的几个数字合成一个数字。最后求平均值。

**【算法描述】**

定义 ifstream 对象并打开输入文件；
j=0；
当(文件未处理完){
　　从输入文件读取一行信息到 char 类型数组 str 中；
　　N=0；　　　　　　　　　　　　　　　　//存储转换后的整数

```
    i=0;
    当 str[i]不是逗号或结束标志时 {
        将 str[i]转换为数字存入 k;
        N=N*10+k;                        //将 k 合成到 N 中
    }
    并存入到 c[j]中;                      //将数字依次存入到 c 数组
    j++;
}
计算 c 数组平均值,输出;
```

## 3.11 习题 11 数据结构、算法与应用

1. 用 vector 建立一个以 string 为元素的向量,编程完成以下几步操作:

① 首先插入元素"China"、"Japan"、"Itali"、"French",显示向量内容。

② 将"American"插入到列表的第二个位置("Japan"之前),显示向量内容。

③ 将元素排序,显示向量内容。

④ 删除元素"Japan",显示向量内容。

**【编程提示】** 向 vector 插入元素可以用 vector 类的 push_back 或 insert 函数,显然第①个操作可以用这两种函数中的任意一种实现,但第②步只能用 insert 函数实现。

排序使用 sort 算法,而删除使用 vector 的 erase 函数。注意,要先找到"Japan"的位置(即迭代器),才能利用 erase 删除元素。

2. 一个四则运算表达式是由数字、运算符和小括号连接起来的。正确的表达式其左右括号数字应相等。比如 3.1+((2.3-1.2)*6-4.2) 或 3.14*(2-1.2)*((5-1.2)/2) 等。假定表达式以字符方式读入,用栈编写一个程序,判断一个表达式的左、右括号是否匹配。

**【编程提示】** 可以依次读取表达式字符串的每个字符,遇到'(',就将其入栈;遇到')',就做一次出栈操作。若最后栈空则括号匹配正确,否则匹配失败。

3. 用栈辅助实现将输入的十进制正整数转换为二进制整数。要求将二进制整数逐位存储在字符数组中,最后在屏幕上输出。

**【问题分析】** 一个十进制整数 N 转换为二进制数的过程如下:首先,用 N 对 2 求余得到二进制的最右边的数字,然后令 N=N/2 取整。若 N 不为零,则继续用 N 对 2 求余得到二进制的从右边数第二位数字。再令 N=N/2 取整,以此类推。当 N 为零时,所有余数从右向左连起来就是结果。

由上面方法得到的二进制数的每一位是从右向左依次得到,不利于按正常顺序将数字存储到数组中;因此可以用栈将得到的余数依次压栈,最后再依次出栈,就可得到正确的二进制数。

**【算法描述】**

读入十进制数 N;

```
当(N 不为 0){
    R=N%2,将 R 入栈;
    N=N/2;
}
将数据依次出栈,并依次存入字符数组 str[]中;
输出 str;
```

4. 编写程序输出集合{1、2、3}的所有非空子集。

**【编程要点】**

用枚举法解决。注意“集合”不是“排列”,集合的元素是无序的。

**【算法描述】**

```
循环(i 从 1 至 3){
    输出集合(i)
    循环(j 从 i+1 至 3){
        输出集合(i,j)
        循环(k 从 j+1 至 3){
            输出集合(i,j,k)
        }
    }
}
```

5. 编写程序实现如下功能:将数字 1~6 填入连续的 6 个方格中,使得相邻的两个数字之和为素数。

**【问题分析】** 本题可以用枚举法解决,也可以用回溯的方法解决。在此给出第二种方法的分析。

首先考察原问题能不能化为若干个解法类似的决策阶段。显然可以认为每次在一个格子中放一个数字是一个决策。这个数字的放置规则是:既不能与前面所有位置的数字重复,还必须与前一个位置的数字之和为质数。于是可以针对在一个格子中放置数字创建一个函数,利用上面的规则确定递归回溯的过程。

**【算法描述】** 假如 6 个格子分别对应一个全局数组 ch 的 0~5 号的下标位置,可以建立一个函数 PutNum( int  n),表示在数组的下标位置 n 上放置数字的方法。PutNum 函数可描述如下:

```
PutNum(int  n)                              //在第 n个位置放置数字
{
    若 n为 6,则                             //找到了一个解
        输出结果;
    否则 {
        循环 (i从 1到 6) {                  //尝试放置数字 i
            若 n为 0,则 ch[0]=i;            //第一个位置直接放
            否则 {
                若 ch[0]至 ch[i-1]与数字 i不重复,且 ch[i-1]+i 为质数,则
```

```
                    {
                        ch[n]=i;
                        PutNum(n+1);                //在第 n+1 个位置放置数字
                    }
                }
            }
        }
    }
```

在主函数中，只要直接调用 PutNum(0)即可。

6. 给定三种物品和一个背包。物品的重量是 18、15、10，其价值为 25、24、15，背包的载重量为 20。在选择物品装入背包时，可以选择物品的一部分，而不一定要全部装入背包。应如何选择装入背包的物品，使得装入背包中物品的总价值最大？

**【问题分析】** 这是一个可以用贪心法得到最优解的问题。首先求出每件物品单位重量的价值，按单位重量的价值从大到小的顺序装入背包。最后一件若装不下，则切取其一部分装入即可。

**【算法描述】** 设物品重量放在数组 w[]中，价值放在数组 v[]中。

① 计算每个 v[i]/w[i]存入 p[i]中。

② 将 p[i]从大到小排序，同时将 w 和 v 数组也调换位置(和 p 数组对应)。

③ 用背包容量逐个减去 w[0]、w[1]等，直到无法减去 w[k]。　//物品 k 无法装入

④ 用剩余容量乘以 p[k]，再加上 v[0]至 v[k−1]就得到最大价值；

7. 跳棋问题

假定跳棋子的走法有平移和跳跃两种。平移是向没有子的上下左右相邻位置移动一格，跳跃是跳过一个有子的位置到达镜像位置的另一边。如果满足条件，则可以连续跳跃。假定有 4×4 的棋盘，棋盘状态由用户输入。0 代表没有子，1 代表有子，2 代表将要移动的棋子。编写程序输出给定位置跳棋子的所有一步可达位置。

示例输入如下：

```
0 0 1 0
0 1 0 0
1 0 0 1
0 1 2 0
```

示例输出如下：

```
0 0 1 0
3 1 3 0
1 0 3 1
3 1 2 3
```

其中，3 表示 2 位置的棋子一步可以到达的所有位置。

**【编程提示】** 棋盘状态显然用二维数组存放。可以用类似于图的广度优先遍历的方法解决本问题。

**【算法描述】**

① 考察平移可以走到的位置,将数组中这些位置设定为 3。

② 考察跳跃可以走到的位置,将数组中这些位置设定为 3,并且将这些位置(比如位置[4,0])放入队列中。

③ 取队列头部元素,考察该位置能跳跃到达的新位置(号码为 0),将新位置设定为 3,并将新位置入队。

④ 反复执行第③步,直到队列中元素为空。

# 第4部分

## 常用资料

## 4.1 ASCII字符表

| 符　　号 | 十进制 | 八进制 | 十六进制 | 符　　号 | 十进制 | 八进制 | 十六进制 |
|---|---|---|---|---|---|---|---|
| NUL 空字符(Null) | 0 | 0 | 0H | | 29 | 35 | 1DH |
| | 1 | 1 | 1H | | 30 | 36 | 1EH |
| | 2 | 2 | 2H | | 31 | 37 | 1FH |
| | 3 | 3 | 3H | 空格符 | 32 | 40 | 20H |
| | 4 | 4 | 4H | ! | 33 | 41 | 21H |
| | 5 | 5 | 5H | " | 34 | 42 | 22H |
| | 6 | 6 | 6H | # | 35 | 43 | 23H |
| BEEP 响铃 | 7 | 7 | 7H | $ | 36 | 44 | 24H |
| 退格 | 8 | 10 | 8H | % | 37 | 45 | 25H |
| '\t'水平制表符 | 9 | 11 | 9H | & | 38 | 46 | 26H |
| '\n'换行 | 10 | 12 | AH | ' | 39 | 47 | 27H |
| '\v'垂直制表符 | 11 | 13 | BH | ( | 40 | 50 | 28H |
| '\f'换页 | 12 | 14 | CH | ) | 41 | 51 | 29H |
| '\r'回车 | 13 | 15 | DH | * | 42 | 52 | 2AH |
| shift out | 14 | 16 | EH | + | 43 | 53 | 2BH |
| shift in | 15 | 17 | FH | , | 44 | 54 | 2CH |
| | 16 | 20 | 10H | - | 45 | 55 | 2DH |
| | 17 | 21 | 11H | . | 46 | 56 | 2EH |
| | 18 | 22 | 12H | / | 47 | 57 | 2FH |
| | 19 | 23 | 13H | 0 | 48 | 60 | 30H |
| | 20 | 24 | 14H | 1 | 49 | 61 | 31H |
| | 21 | 25 | 15H | 2 | 50 | 62 | 32H |
| | 22 | 26 | 16H | 3 | 51 | 63 | 33H |
| | 23 | 27 | 17H | 4 | 52 | 64 | 34H |
| 取消 | 24 | 30 | 18H | 5 | 53 | 65 | 35H |
| | 25 | 31 | 19H | 6 | 54 | 66 | 36H |
| | 26 | 32 | 1AH | 7 | 55 | 67 | 37H |
| ESC | 27 | 33 | 1BH | 8 | 56 | 70 | 38H |
| | 28 | 34 | 1CH | 9 | 57 | 71 | 39H |

续表

| 符　　号 | 十进制 | 八进制 | 十六进制 | 符　　号 | 十进制 | 八进制 | 十六进制 |
|---|---|---|---|---|---|---|---|
| ： | 58 | 72 | 3AH | ] | 93 | 135 | 5DH |
| ； | 59 | 73 | 3BH | ^ | 94 | 136 | 5EH |
| ＜ | 60 | 74 | 3CH | _ | 95 | 137 | 5FH |
| ＝ | 61 | 75 | 3DH | ` | 96 | 140 | 60H |
| ＞ | 62 | 76 | 3EH | a | 97 | 141 | 61H |
| ? | 63 | 77 | 3FH | b | 98 | 142 | 62H |
| @ | 64 | 100 | 40H | c | 99 | 143 | 63H |
| A | 65 | 101 | 41H | d | 100 | 144 | 64H |
| B | 66 | 102 | 42H | e | 101 | 145 | 65H |
| C | 67 | 103 | 43H | f | 102 | 146 | 66H |
| D | 68 | 104 | 44H | g | 103 | 147 | 67H |
| E | 69 | 105 | 45H | h | 104 | 150 | 68H |
| F | 70 | 106 | 46H | i | 105 | 151 | 69H |
| G | 71 | 107 | 47H | j | 106 | 152 | 6AH |
| H | 72 | 110 | 48H | k | 107 | 153 | 6BH |
| I | 73 | 111 | 49H | l | 108 | 154 | 6CH |
| J | 74 | 112 | 4AH | m | 109 | 155 | 6DH |
| K | 75 | 113 | 4BH | n | 110 | 156 | 6EH |
| L | 76 | 114 | 4CH | o | 111 | 157 | 6FH |
| M | 77 | 115 | 4DH | p | 112 | 160 | 70H |
| N | 78 | 116 | 4EH | q | 113 | 161 | 71H |
| O | 79 | 117 | 4FH | r | 114 | 162 | 72H |
| P | 80 | 120 | 50H | s | 115 | 163 | 73H |
| Q | 81 | 121 | 51H | t | 116 | 164 | 74H |
| R | 82 | 122 | 52H | u | 117 | 165 | 75H |
| S | 83 | 123 | 53H | v | 118 | 166 | 76H |
| T | 84 | 124 | 54H | w | 119 | 167 | 77H |
| U | 85 | 125 | 55H | x | 120 | 170 | 78H |
| V | 86 | 126 | 56H | y | 121 | 171 | 79H |
| W | 87 | 127 | 57H | z | 122 | 172 | 7AH |
| X | 88 | 130 | 58H | { | 123 | 173 | 7BH |
| Y | 89 | 131 | 59H | \| | 124 | 174 | 7CH |
| Z | 90 | 132 | 5AH | } | 125 | 175 | 7DH |
| [ | 91 | 133 | 5BH | ～ | 126 | 176 | 7EH |
| \ | 92 | 134 | 5CH | 删除 | 127 | 177 | 7FH |

## 4.2 Visual C ++ 编译错误中的常见词汇

access 存取
allocate 分配,分派
ambiguous 模棱两可的,不清楚的
array 数组
assignment 赋值语句
assume 假设
available 可用的
character 字符
class 类
compiling 编译
configuration 配置
constant size 常量大小
constant 常量
constructor 构造函数
control 控制
conversion n. 变换,转换
convert v. 变换,转换
debug 调试
declaration n. 声明
declare 声明
destructor 析构函数
differs 不同,有异
directory 目录
effect 效果,有效
error 错误
executing 执行
expect 期待,盼望
expression 表达式
external 外部的
follow 跟随
function 函数
function-style cast 函数样式(形式)
header file 头文件
identifier 标识符
illegal 不合法的
indirection 间接
initialize vt. 初始化
local variable 局部变量
l-value 左值
member 成员
mismatch 不匹配
missing 丢失的
object 对象
operand 操作数
operator 操作符
overload 重载
parameter 参数
path 路径
perform 完成
pointer 指针
predefined 预定义的
private 私有
redefinition 再定义,重复定义
reference 引用,使用
reinterpret 重新解释
require 需要
resolution 解决(方案)
return 返回
specifier 区分符,分类符
storage-class 存储类
subscript 下标,下标的
syntax 语法
tag 标签,标记
type 类型
undeclared 未声明的
unknown size 未知的大小
unknown 未知的,不知道的
unreferenced 未被引用的
unresolved 无法解决的
user-defined-conversion 用户定义的

变换

value　数值

variable　变量

warning　警告

## 4.3　Visual C++　6.0 编程环境下常见的编译错误

以下按错误的代码排序。

(1) fatal error C1004：unexpected end of file found

遇到了不该遇到的文件尾。(一般是大括号不匹配，或工程类型不正确)

(2) fatal error C1083：Cannot open include file：'a. h'：No such file or directory

不能打开包含文件“a. h”：没有这样的文件或目录。

(3) error C2018：unknown character '0xa3'

error C2018：unknown character '0xbb'

不认识的字符'0xa3'和'0xbb'(一般是汉字或中文标点符号造成的，应改为英文符号)。

(4) error C2106：'='：left operand must be l－value

运算符的左边应该是一个“左值”，即变量。

(5) error C2065：'d'：undeclared identifier

“d”：未声明过的标识符。

(6) error C2087：'<Unknown>'：missing subscript

二维数组作为函数的参数时，必须指定第二个下标的最大数。

(7) error C2133：'x'：unknown size

数组 x 未知大小。

(8) error C2143：syntax error：missing ';' before '}'

句法错误：“}”前缺少“;”。

(9) error C2248：'x'：cannot access private member declared in class 'A'

“x”是 A 类的私有成员，外部不可访问。

(10) error C2371：'a'：redefinition

标识符“a”重定义。

(11) error C2512：'A'：no appropriate default constructor available

A 类无缺省的构造函数。

(12) error C2676：binary '<<'：'class std::basic_istream<char,struct std::char_traits<char>>' does not define this operator or a conversion to a type acceptable to the predefined operator

“<<”符号用反或用错。

(13) error C2676：binary '>>'：'class std::basic_ostream<char,struct std::char_traits<char>>' does not define this operator or a conversion to a type acceptable to the predefined operator

“>>”符号用反或用错。

(14) warning C4101：'a2'：unreferenced local variable

“a2”未引用过的局部变量。

(15) warning C4700：local variable 'a' used without having been initialized

局部变量“a”未初始化。

(16) warning C4715：'Max'：not all control paths return a value

函数中不是全部路径都返回了一个值。

(17) error C4716：'Max'：must return a value

函数“Max”必须返回一个值。

(18) fatal error LNK1168：cannot open Debug/P1.exe for writing

连接错误：不能打开 P1.exe 文件，以改写内容。（一般是 P1.Exe 还在运行，未关闭。关闭程序的运行窗口）

(19) error LNK2001：unresolved external symbol " double __cdecl Max (double，double，int)"  (? Max@@YANNNH@Z)

连接错误，函数“Max”声明、定义和调用不一致。请检查声明、定义和调用的函数名。

(20) error LNK2001：unresolved external symbol _WinMain@16

工程类型不正确，应为“Win32 Console Application”。

(21) error LNK2001：unresolved external symbol _main

连接错误，主函数不存在。请检查 main 函数的名称是否正确，不要写成“mian”。

## 4.4　常用数学库函数

下面的函数包含在 cmath(或 math.h)头文件中。

(1) int abs(int i)，返回整型参数 i 的绝对值；

(2) double fabs(double x)，返回双精度参数 x 的绝对值；

(3) long labs(long n)，返回长整型参数 n 的绝对值；

(4) double exp(double x)，返回指数函数 $e^x$ 的值；

(5) double log(double x)，返回 $\ln_e^x$ 的值；

(6) double log10(double x)，返回 $\log_{10}^x$ 的值；

(7) double pow(double x,double y)，返回 $x^y$ 的值；

(8) double pow10(int p)，返回 $10^p$ 的值；

(9) double sqrt(double x)，返回 $+\sqrt{x}$ 的值；

(10) double acos(double x)，返回 x 的反余弦 $\cos^{-1}(x)$ 值，x 为弧度，x 的定义域为 $[-1.0, 1.0]$，值域为 $[0, \pi]$；

(11) double asin(double x)，返回 x 的反正弦 $\sin^{-1}(x)$ 值，x 为弧度，x 的定义域为 $[-1.0, 1.0]$，值域为 $[-\pi/2, +\pi/2]$；

(12) double atan(double x)，返回 x 的反正切 $\tan^{-1}(x)$ 值，x 为弧度，值域为 $(-\pi/2, +\pi/2)$；

(13) double cos(double x)，返回 x 的余弦 cos(x)值，x 为弧度；

(14) double sin(double x),返回 x 的正弦 sin(x)值,x 为弧度;

(15) double tan(double x),返回 x 的正切 tan(x)值,x 为弧度;

(16) double cosh(double x),返回 x 的双曲余弦值,x 为弧度,$\cosh(x)=(e^{x}+e^{-x})/2$;

(17) double sinh(double x),返回 x 的双曲正弦值,x 为弧度,$\sinh(x)=(e^{x}-e^{-x})/2$;

(18) double tanh(double x),返回 x 的双曲正切值,x 为弧度,$\tanh(x)=(e^{x}-e^{-x})/(e^{x}+e^{-x})$;

(19) double ceil(double x),返回不小于 x 的最小整数;

(20) double floor(double x),返回不大于 x 的最大整数;

(21) void srand(unsigned seed),初始化随机数发生器,常见用法是:srand(time(NULL));//需另包含头文件 time.h 或 ctime;

(22) int rand(),返回一个[0, RAND_MAX]内的随机数。RAND_MAX 的值可以通过 cout<< RAND_MAX;观察到。

## 4.5 常用的字符串处理函数

下列函数包含在头文件 cstring(或 stirng.h)中。

(1) char * strcat(char * dest,const char * src),将字符串 src 添加到 dest 末尾,返回 dest。注意,dest 应有足够的空间再容纳 src。

(2) char * strchr(const char * s,int c),检索并返回字符 c 在字符串 s 中第一次出现的位置(内存地址,指针),如果找不到匹配字符,返回 NULL。

(3) int strcmp(const char * s1,const char * s2),比较字符串 s1 与 s2 的大小,并返回 s1－s2 (结果大于 0 表示 s1 大于 s2;结果小于 0 表示 s1 小于 s2;结果等于 0 表示它们相同)。

(4) char * strcpy(char * dest,const char * src),将字符串 src 复制到 dest,返回 dest。

举例:

```
#include <iostream>
using namespace std;
#include<cstring>
void main()
{
    char str1[100];                         //定义字符数组,存放字符串
    char str2[100];
    char str3[80]="Li Xian crashes out of men's 110m hurdles 2012-08-07.";
    strcpy(str1,str3);                      //用法一,函数语句
    cout<<str1<<endl;;                      //
    cout<<strcpy(str2,str3)<<endl;          //用法二,函数作表达式的一部分
    cout<<str2<<endl;;                      //
```

```
}
```

运行结果：

```
Li Xian crashes out of men's 110m hurdles 2012-08-07.
Li Xian crashes out of men's 110m hurdles 2012-08-07.
Li Xian crashes out of men's 110m hurdles 2012-08-07.
```

(5) size_t strcspn(const char * s1,const char * s2)，扫描 s1，返回在 s1 中有，在 s2 中也有的第一个字符的下标。

(6) int stricmp(const char * s1,const char * s2)，比较字符串 s1 和 s2，不区分大小写，返回 s1－s2，不改变 s1 和 s2。

(7) size_t strlen(const char * s)，返回字符串 s 的长度。

(8) char * strncat(char * dest,const char * src,size_t maxlen)，将字符串 src 中最多 maxlen 个字符连接到字符串 dest 后面，返回 dest。

(9) int strncmp(const char * s1,const char * s2,size_t maxlen)，比较字符串 s1 与 s2 中的前 maxlen 个字符(至多)，s1＞s2 返回正数，s1＜s2 返回负数，s1＝s2 返回 0。

(10) char * strncpy(char * dest,const char * src,size_t maxlen)，复制 src 中的前 maxlen(至多)个字符到 dest 中，返回 dest。

(11) int strnicmp(const char * s1,const char * s2,size_t maxlen)，不区分大小写地比较字符串 s1 与 s2 中的至多前 maxlen 个字符。

(12) char * strrchr(const char * s,int c)，返回字符 c 在字符串 s 中最后一次出现的位置(指针)，如果找不到，返回 NULL。

(13) size_t strspn(const char * s1,const char * s2)，返回 s1 中第 1 个由 s2 中的字符组成的字符串的长度，或者说返回 s1 中第 1 个不是 s2 中的字符的字符的下标。

如 s1[]＝"sfdddxyzabfcabcffff",s2[]＝"fsabfcd"，则结果为 5。

(14) char * strstr(const char * s1,const char * s2)，在字符串 s1 中查找字符串 s2，找到则返回第一次出现的位置(指针)，找不到则返回 NULL。

(15) char * strlwr(char * s)，将字符串 s 中的大写字母全部转换成小写字母，返回 s。

(16) char * strupr(char * s)，将字符串 s 中的小写字母全部转换成大写字母，返回 s。

(17) char * strrev(char * s)，将字符串 s 中的字符全部颠倒顺序重新排列，返回 s。

(18) char * strset(char * s,int ch)，将字符串 s 中的所有字符替换成给定字符 ch。

## 4.6 常用字符串和数的转换函数

下列函数包含在 cstdlib(或 stdlib.h)中。

(1) double atof(char * nptr)，将字符串 nptr 转换成浮点数并返回，错误返回 0。

(2) double atoi(char * nptr)，将字符串 nptr 转换成整数并返回，错误返回 0。

(3) double atol(char * nptr),将字符串 nptr 转换成长整数并返回,错误返回 0。

(4) char * ecvt(double value,int ndigit,int * decpt,int * sign),将浮点数 value 转换成字符串并返回该字符串,其中 value 是待转换的双精度数,ndigit 是转换后保留的数字位数,decpt 是输出参数,保存双精度数的整数位数,sign 是输出参数,保留双精度数的符号。例如:

```
#include <iostream>
using namespace std;
#include<cmath>
void main()
{
    int      decimal,   sign;
    char     * buffer;                               //
    int      precision=8;
    double   source=-13.1415926535;
    buffer=ecvt(source, precision, &decimal, &sign);  //转换
    cout.setf (ios::fixed);cout.precision(10);        //显示双精度数小数点后 10 位
    cout<<source<<endl;                               //原数
    cout<<buffer<<endl;                               //precision=8;保留 8 位
    cout<<decimal<<endl;                              //整数部分 13,两位
    cout<<sign<<endl;                                 //负数 1,整数 0
}
```

运行结果为:

```
-13.1415926535
13141593
2
1
```

(5) char * itoa(int value,char * string,int radix) 将整数 value 转换成字符串存入 string,radix 为转换时所用基数。例如:

```
#include <iostream>
using namespace std;
#include<cmath>
void main()
{
    char buffer[20];
    int  i=3445;
    itoa(i, buffer, 16);                              //以 16 为基数转换为字符串
    cout<<i<<"   "<<buffer<<endl;                     //显示
    itoa(i, buffer, 2  );                             //以 2 位基数转换为字符串
    cout<<i<<"   "<<buffer<<endl;                     //显示
}
```

运行结果为：

```
3445    d75
3445    110101110101
```

## 4.7　string 类的常用方法

使用 string 类，需要包含头文件 string。

### 1. string 类的构造函数

（1）string(const char * s)；　//用 c 字符串 s 初始化，例如 string　str1("Olympic 2012")；

（2）string(int n,char c)；　//用 n 个字符 c 初始化，如 string str2(10，'v')；

此外，string 类还支持默认构造函数和复制构造函数，如 string s1；string s2＝"hello"；都是正确的写法。当构造的 string 太长而无法表达时会抛出 length_error 异常。

### 2. string 类对象的输入输出操作

（1）string 类重载运算符“＞＞”用于输入，重载运算符“＜＜”用于输出操作，如 string s；cin＞＞s;cout＜＜s;。

（2）函数 getline(istream &in,string &s)；用于从输入流 in 中读取字符串到 s 中，以换行符'\n'分开，例如 string s;getling(cin,s);cout＜＜s;。

### 3. string 类对象可直接使用的运算符

string 类重载了＝、＋、＋＝、＞、＜、＞＝、＜＝、＝＝、!＝、[]等运算符。

（1）＝，赋值，如 string s1,s2;s1 ＝"downpour",s2＝s1;。

（2）＋、＋＝是连接运算。

（3）＞、＜、＞＝、＜＝、＝＝、!＝是比较运算。

（4）还可以使用 assign()、append()、compare()等成员函数实现赋值、连接和比较。

（5）[]，取指定下标的字符，原形为：char &operator[](int n);//返回第 n 个字符（从 0 开始），也可以出现在等号的左边。例如 string　str1("Olympic 2012")；cout＜＜str1[2]；结果为 y。str1[2]＝Y;原小写 y 改为大写 Y。

### 4. 反映 string 的特征的成员函数：

（1）int capacity()const；//返回当前容量（即 string 中不必增加内存即可存放的元
//素个数）

（2）int max_size()const；//返回 string 对象中可存放的最大字符串的长度

（3）int size()const；　　//返回当前字符串的大小

(4) int length()const;　　//返回当前字符串的长度

(5) bool empty()const;　　//判断当前字符串是否为空

(6) void resize(int len,char c);　　//把字符串当前大小置为 len,多余的部分删除,//如果不足用字符 c 填充

**5. string 类的字符操作(复制、查找、插入、替换、取子串等)成员函数**

(1) char &at(int n);　//返回第 n 个字符(从 0 开始),例如 string　str1("Olympic //2012");cout<<str1.at(2);结果为 y

(2) const char *data()const;　//返回一个非 null 终止的 char 型字符数组

(3) const char *c_str()const;　//返回一个以 null 终止的 char 型字符串

(4) int copy(char *s, int n, int pos=0)const;　//把当前串中以 pos 开始的 n 个//字符拷贝到以 s 为起始位置的字符数组中,返回实际拷贝的数目

(5) string substr(int pos=0, int n=npos)const;　//返回 pos 开始的 n 个字符组//成的字符串

(6) void swap(string &s2);　//交换当前字符串与 s2 的值

(7) int find(char c, int pos=0)const;　//从 pos 开始查找字符 c 在当前字符串的//位置

(8) int find(const char *s, int pos=0)const;　//从 pos 开始查找字符串 s 在当前//串中的位置

(9) int find(const string &s, int pos=0)const;　//从 pos 开始查找字符串 s 在当//前串中的位置

(10) string &replace(int p0, int n0,const char *s);　//删除从 p0 开始的 n0 个字//符,然后在 p0 处插入串 s

(11) string &replace(int p0, int n0,const string &s);　//删除从 p0 开始的 n0 个//字符,然后在 p0 处插入串 s

(12) string &insert(int p0, const char *s);　//在 p0 处插入字符串 s

(13) string &insert(int p0,const string &s);　//在 p0 处插入字符串 s

(14) string &insert(int p0, int n, char c);　//此函数在 p0 处插入 n 个字符 c

(15) string &erase(int pos=0, int n=npos);　//删除 pos 开始的 n 个字符,返回//修改后的字符串

# 参 考 文 献

[1] 赵英良,夏秦,仇国巍等. 大学计算机基础.第 4 版. 北京：清华大学出版社，2011
[2] 苏小红,王宇颖等. C 语言程序设计. 北京：高等教育出版社,2011
[3] 罗建军等. C/C++ 语言程序设计案例教程. 北京：清华大学出版社，2010
[4] 郑莉,董渊,何江舟等. C++ 语言程序设计.第 4 版. 北京：清华大学出版社,2010
[5] 钱能.C++ 程序设计教程(修订版)——设计思想与实现. 北京：清华大学出版社，2009
[6] 孙淑霞,李思明等. C/C++ 程序设计实验指导与测试. 北京：电子工业出版社，2009
[7] 罗建军,朱丹军,顾刚等. C++ 程序设计教程.第 2 版. 北京：高等教育出版社,2007
[8] 罗建军,朱丹军等.C++ 程序设计教程学习指导.第 2 版. 北京：高等教育出版社,2007
[9] 吴乃陵，况迎辉. C++ 程序设计.第 2 版. 北京：高等教育出版社,2006
[10] 吴乃陵,李海文. C++ 程序设计实践教程.第 2 版. 北京：高等教育出版社,2006